UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

INGENIERÍA DE SISTEMAS

Tesis previa a la obtención del título de: INGENIERO DE SISTEMAS

TEMA:
ANÁLISIS, DISEÑO, DESARROLLO E INTEGRACIÓN DEL MÓDULO GESTIÓN DE HORARIOS DEL SISTEMA UPS SCHEDULE PARA LA UNIVERSIDAD POLITÉCNICA SALESIANA SEDE QUITO

AUTORES:
OVIEDO GUALLICHICO DANILO ROMÁN
GRANDA LEÓN JOSÉ ANTONIO

DIRECTOR:
RODRIGO EFRAÍN TUFIÑO CÁRDENAS

Quito, febrero de 2014

DECLARATORIA DE RESPONSABILIDAD Y AUTORIZACIÓN DE USO DEL TRABAJO DE TITULACIÓN

Nosotros Danilo Román Oviedo Guallichico y José Antonio Granda León autorizamos a la Universidad Politécnica Salesiana la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

Además declaramos que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad de los autores.

_____

Danilo Román Oviedo Guallichico
CC: 171818469 8

_____

José Antonio Granda León
CC: 1717602609

ÍNDICE

ÍNDICE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La generación de horarios en la Universidad Politécnica Salesiana ha sido una problemática latente, el uso de recurso humano, físico y tiempo han hecho de este proceso una complicación período tras período, debido a la falta de procesos definidos y herramientas tecnológicas que faciliten esta tarea.

En la búsqueda de resolver este problema se plantea la elaboración del sistema "UPS Schedule", el mismo que fue dividido en varios módulos por su complejidad. Partiendo de esta premisa, el proyecto se enfoca en el "módulo de Gestión de Horarios" sin dejar de lado el trabajo conjunto entre los integrantes del sistema "UPS Schedule" en la definición de tecnología, modelamiento y estructura.

Tras lo expuesto, la elaboración del "módulo de Generación de Horarios" usa nuevas tecnologías que permiten una interacción con el usuario de forma ágil, eficiente e intuitiva en búsqueda de aceptación por parte de los involucrados en la generación de horarios.

Haciendo uso de una metodología bien formada como OMT, se logra entender y modelar el problema y la solución, dando como resultado un módulo que mejora de forma notable el proceso de generación de horarios.

ABSTRACT

Schedules generation at Salesian Polytechnic University has been a latent problem, the use of human, physical and time resources have made this process a complication period after period, due to the lack of defined processes and technological tools to facilitate this task.

In searching to solve this problem the development of the " UPS Schedule" system, it was divided into modules. On this basis, the project focuses on the "modulo de Gestión de Horarios" without neglecting the joint work of the members of the " UPS Schedule" system in the definition of technology, modeling and structure.

Following the above, the development of "módulo de Generación de Horarios" uses new technologies that enable user interaction in a flexible, efficient and intuitive seeking acceptance from those involved in generating schedules.

Using a methodology OMT well shaped as, one can understand and model the problem and the solution, resulting in a module that significantly improves the process of generating schedules.

INTRODUCCIÓN

En el presente trabajo de tesis se da solución a la problemática en la generación de horarios de la Universidad Politécnica Salesiana Sede Quito, mediante el desarrollo del módulo de generación de horarios, utilizando herramientas de licencia GNU y nuevas tecnologías que permiten la interacción de los usuarios de manera amigable, sencilla e intuitiva.

En la búsqueda de solucionar el problema se realiza el análisis de la situación actual, mediante la metodología OMT se logra modelar dicho problema, tras el análisis y varios procesos de iteración que definen el modelo de la solución.

Solución que fue desarrollada bajo la perspectiva multiplataforma para evitar las limitantes de Sistema Operativo y sencilla para la integración, mantenimiento y nuevos desarrollos.

CAPÍTULO 1
PLANTEAMIENTO DEL PROBLEMA

La tarea de realizar horarios académicos se ha vuelto muy compleja, puesto que se ha incrementado el número de carreras, docentes (que dictan clases en más de una carrera) y el número de cursos ofertados, los mismos que varían de período a período.

Otro factor a considerar es la disponibilidad del docente, debido a que muchos de ellos tienen asignados otras tareas (investigación, gestión administrativa, dirección de tesis, etc.) o trabajan en instituciones externas a la Universidad Politécnica Salesiana.

Los directores de las carreras, utilizan tiempo en exceso para la generación de horarios. Ya que, las herramientas actuales no permiten manejar históricos de horarios generados en períodos anteriores, forzando a ser elaborados sin una base inicial, a pesar de que en ocasiones la variación entre períodos es mínima.

Los docentes que dictan cátedras en más de una carrera, tienen problemas debido a que los horarios son generados en base a la visión de una sola carrera, lo que ocasiona conflictos por cruce de horarios en otras carreras y a la vez forzando a los directores de carrera a utilizar recursos no planificados para volver a generar horarios.

Si al uso limitado de recursos como: tiempo, conocimiento y humano; se agrega cambios de última hora, genera en el director de carrera un gran nivel de estrés y agotamiento, mismos que lo desvirtúan de las actividades que este debe realizar.

Es importante mencionar que la falta de recursos de comunicación, incrementa notablemente los tiempos para la generación de horarios, ya que al existir algún inconveniente como cruce de horarios, cambio o asignación de una materia se entra en una negociación entre docente y director, por lo general vía telefónica o correo para indagar la disponibilidad del docente.

1.1. Objetivos

1.1.1. Objetivo general

Obtener un módulo funcional para gestión de horarios que trabaje de forma integrada al "Sistema UPS Schedule de la Universidad Politécnica Salesiana Sede Quito" (Tufiño, 2012).

1.1.2. Objetivos específicos

- Analizar el proceso de gestión de horarios.
- Diseñar un módulo de gestión de horarios en base a la arquitectura del sistema.
- Programar un módulo de gestión de horarios utilizando herramientas de software libre.
- Permitir la gestión de horarios.
- Validar cruces.
- Imprimir y exportar horarios en formato de fácil manipulación.
- Integrar el módulo al "Sistema UPS Schedule para la Universidad Politécnica Salesiana Sede Quito".
- Probar el funcionamiento del módulo y la correcta interacción con el resto del "Sistema UPS Schedule para la Universidad Politécnica Salesiana Sede Quito".

1.2. Justificación del Proyecto

El propósito de este proyecto es optimizar tiempo, esfuerzo y recursos mediante el desarrollo de un módulo de gestión de horarios para el "Sistema UPS Schedule para la Universidad Politécnica Salesiana Sede Quito" y evitar problema de cruces de horarios (disponibilidad de profesor, cruce entre carreras y cruce en la misma carrera), sobrecarga de trabajo a directores de carrera, disminución de tiempo en la creación, administración y sociabilización de horarios. Garantizando que sea de fácil uso, intuitivo y con alertas de cruces. En busca de un mejor desempeño al momento de realizar los horarios.

El uso de herramientas de software libre, disminuye notablemente los costos de licenciamiento que generaría el desarrollo de este proyecto, pero existe un factor que lo hace novedoso, gracias a que trabajará haciendo uso del desarrollo comunitario con los integrantes de los módulos del "Sistema UPS Schedule para la Universidad Politécnica Salesiana Sede Quito", el cual permitirá que todos los participantes aporten y comuniquen: avances, problemas, modificaciones y cambios para una mejor integración.

1.3. Alcance del Proyecto

Se procederá a diseñar y desarrollar un módulo de gestión de horarios para el "Sistema UPS Schedule para la Universidad Politécnica Salesiana Sede Quito", que deberá ser basado en una plataforma Web, bajo la arquitectura MVC (Modelo Vista Controlador) y con conexión a base de datos.

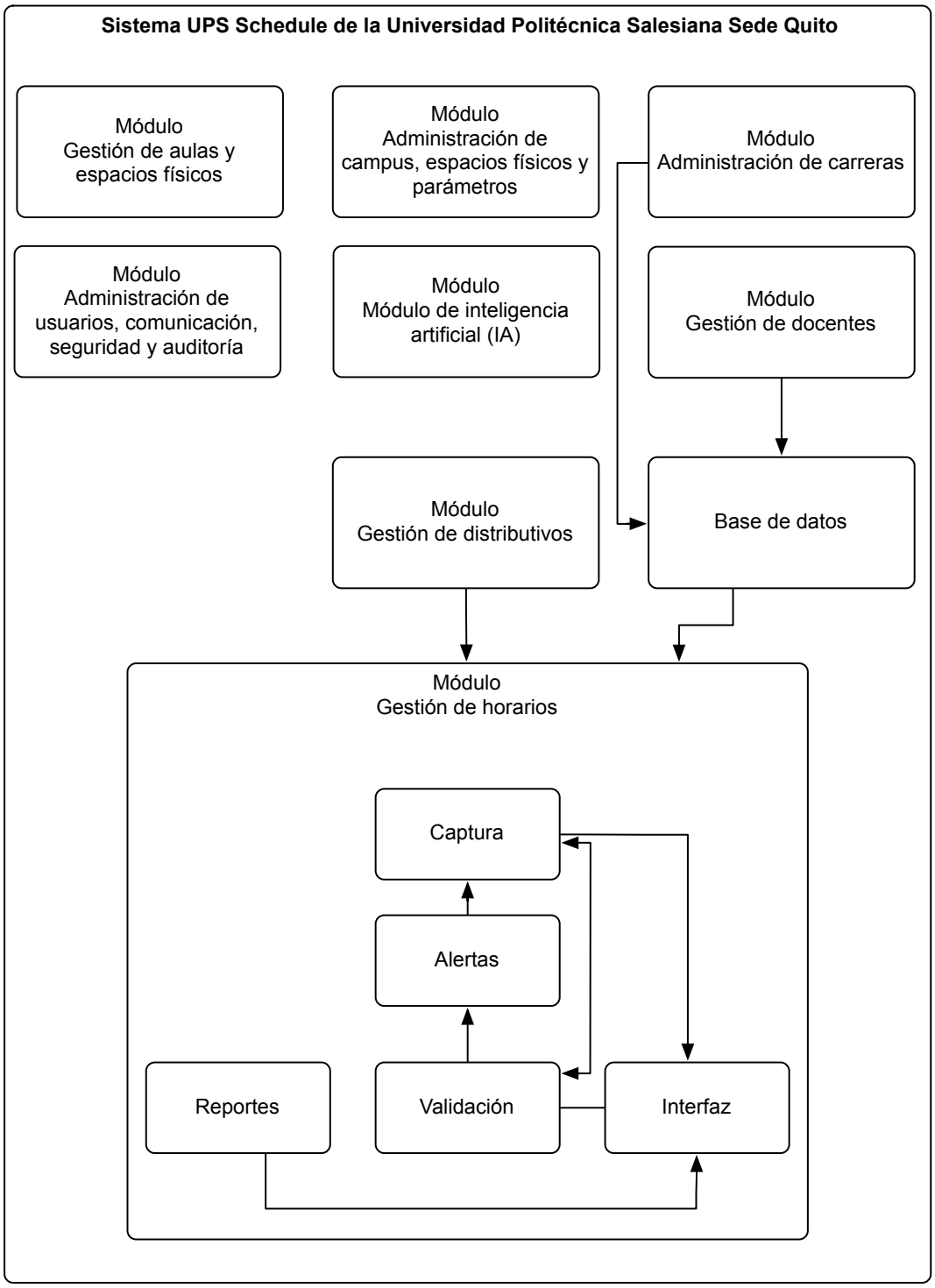
Se hará uso de herramientas tecnologías de software libre como: JAVA, PostgresSQL, SQL Power Architect, GlashFish, PrimeFaces, Netbeans, Jasper Report, LibreOffice y Planner que permitirá a los directores de la Universidad Politécnica Salesiana sede Quito generar, modificar y exportar los horarios generados.

Se establecerá un estándar de nomenclatura y etiquetado para el uso de clases, métodos, variables, tablas y campos que será adoptado por todos los desarrolladores en cada uno de los módulos del "Sistema UPS Schedule de la Universidad Politécnica Salesiana Sede Quito".

El uso de SourceForge buscará una comunicación continua, base del desarrollo comunitario, con la creación de foros, wikis, blogs y discusiones. Haciendo uso de sus repositorios para la distribución de las diferentes versiones de código.

El módulo de gestión de horarios forma parte del "Sistema UPS Schedule de la Universidad Politécnica Salesiana Sede Quito" y se integrará con el módulo "gestión de distributivos" (Tufiño, 2012, pp. 4-6) como se observa a continuación en la figura 1.

Figura 1: Módulos “Sistema UPS Schedule de la Universidad Politécnica Salesiana Sede Quito” con relación al módulo de gestión de horarios

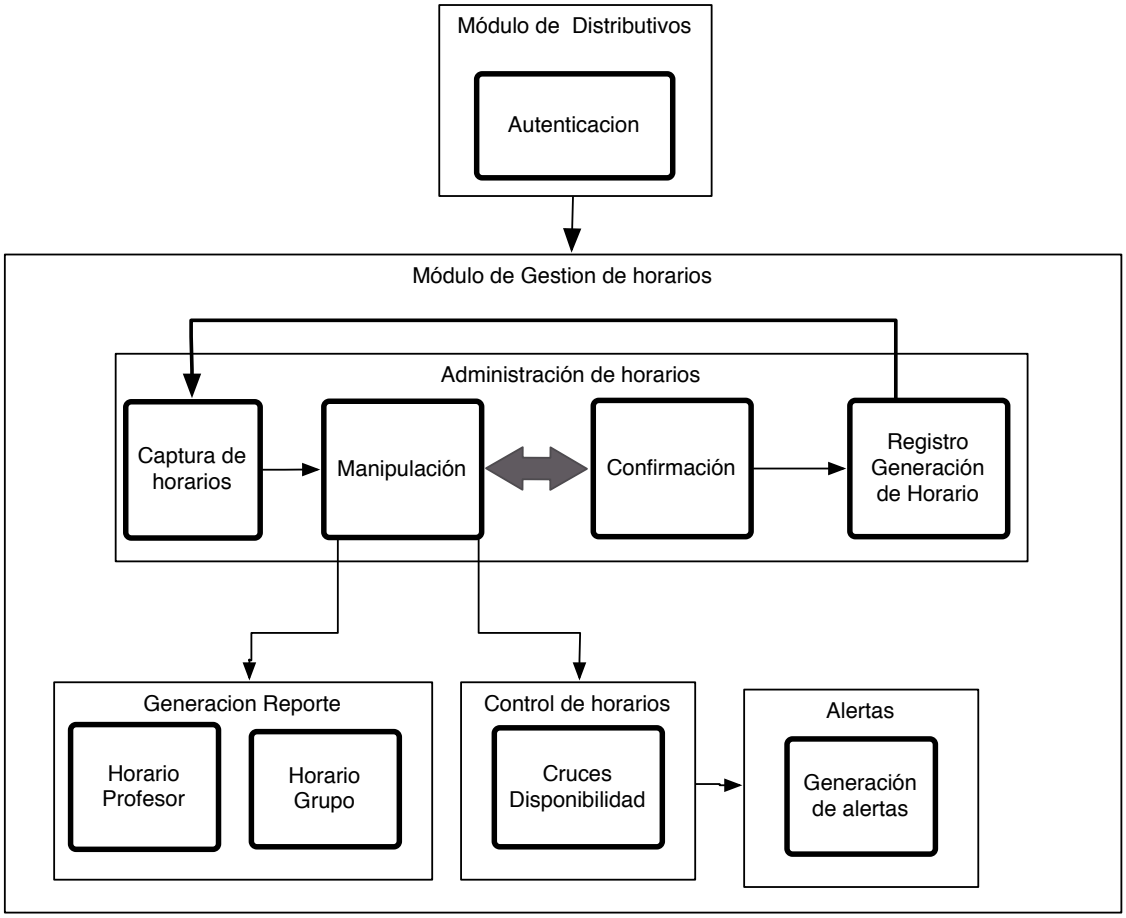


Fuente: Tesistas “Sistema UPS Schedule de la Universidad Politécnica Salesiana Sede Quito”

El módulo de gestión de horarios se encontrará habilitado únicamente para los directores de carrera, trabajando de forma efectiva con una concurrencia de 20

usuarios, pudiendo realizar actividades como: generar nuevo horario por período académico, modificar horario de período actual, exportar horarios como se puede observar en el mapa de procesos de la Figura 2.

Figura 2: Mapa de procesos del módulo de gestión de horarios.



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Las actividades mencionadas anteriormente se realizarán de la siguiente forma: Se notificará a los directores de carrera, mediante alertas, los cambios que desean realizar a un horario, ya que el módulo de generación de horarios validará cruces de carrera, nivel, grupo, materia, horario y disponibilidad; esto mediante una interfaz que manejará tecnologías actuales de desarrollo de sistema Web permitiendo que el uso sea intuitivo y agradable en base a las tendencias de diseño y funcionabilidad que arrojan los últimos estudios del mercado; permitiendo a los directores de carrera gestionar los horarios de una forma rápida, ágil y eficaz.

También permitirá generar reportes en formato PDF, válido para la visualización en herramientas de software libre.

Al ser el módulo de generación de horarios parte del "Sistema UPS Schedule de la Universidad Politécnica Salesiana Sede Quito" es necesario que la arquitectura de la base de datos sea única para todos los módulos.

El proyecto de tesis culminará con la integración del módulo de gestión de horarios al "módulo de gestión de distributivos" mas no con la implementación y capacitación del "Sistema UPS Schedule de la Universidad Politécnica Salesiana Sede Quito".

1.4. Generación de horarios

El proceso de generación de horarios dentro de la Universidad Politécnica Salesiana es complejo, al igual que para varios centros de educación superior, ya que requiere de un metódico y complejo análisis de circunstancias que afectan directamente al normal desarrollo de un período académico. Es por esto que se va a centrar en analizar el proceso actual de generación de horarios, para identificar los problemas que dificultan esta labor.

Para tema de análisis de generación de horarios se va a tomar como referencia la sede Quito, campus SUR y el GIRÓN de la Universidad Politécnica Salesiana.

1.4.1. Proceso de generación de horarios

Esta actividad no se encuentra definida de forma clara como un proceso regulatorio por parte de la universidad, ya que deja a libre elección del director de carrera la generación de horarios. En base a esto se han agrupado a las diferentes formas que los directores de carrera usan para generar sus horarios, en 3 métodos claramente definidos.

1.4.1.1. Proceso manual

Esta forma de generar horarios es una de las más complejas, ya que requiere de la definición clara de varios aspectos como disponibilidad de profesor, definición de materias, cantidad de asistentes y adicional un gran espacio de trabajo. Como se puede observar en la figura 3.

Figura 3: Fotografía del proceso de generación de horarios totalmente manual

Fuente: Nihon Yûkôkai - Amigos de Japón

El proceso inicia con horarios en blanco previamente diseñados en papel, el cual contiene los días de actividad y las horas necesarias. Un horario por cada grupo, luego se coloca sobre este horario las materias con la asignación respectiva de profesor tratando de:

- Asignar las horas correspondientes por semana.
- Evitar colocar a un profesor fuera de su horario disponible.
- Evitar asignar una materia (del profesor) en el mismo día y hora en otro grupo (Tufiño, 2012, pp. 4-6).

Este proceso se repite por cada materia, en ocasiones tomando como base el horario del semestre anterior.

1.4.1.2. Proceso a través de hojas de cálculo

Esta forma de generar horarios es bastante similar a la generación manual, con una diferencia, en este caso los directores de carrera usan como base una hoja de cálculo.

En esta hoja de cálculo diseñan el horario con días y horas necesarias para la carrera y colocan un encabezado con el detalle de la carrera, nivel, grupo, entre otros, como se muestra en la figura 4.

Figura 4: Formato de horario de la carrera de ingeniería civil.

	A	B	C	D	E	F
1		INGENIERIA CIVIL		(SEXTO 2012-2013)		
2	HORARIO:		MATUTINO			
3	AULA: G-16					
4	HORAS	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
5						
6						
7						
8						
9						

Fuente: Director de la carrera de ingeniería civil

En cada celda, los directores de carrera colocan una materia con su respectivo profesor y generan hojas de cálculo distintas por cada nivel, para luego navegar entre hojas o desplazarse dentro de la misma hoja de cálculo e identificar cruces y/o disponibilidad del profesor.

Figura 5: Imagen de un horario de la carrera de ingeniería civil

INGENIERIA CIVIL . PRIMER NIVEL "3"
HORARIO: VESPERTINO
2012-2013 **AULA: A-18**

HORAS	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SABADO	
						INGLES I	7H:00-9H:00
11H:30-13H:30		INGLES I	TÉCNICAS DE INVESTIGACIÓN Lcda.Enith Marcillo			TOPOGRAFIA Ing.Luis Barros	9H:00-11H:00
13H:30 a 15H:30	CALCULO DIFERENCIAL Mat. Luis Felipe Ortega	ESTATICA Ing.Fernando Ulloa	CALCULO DIFERENCIAL Mat. Luis Felipe Ortega	QUÍMICA DE MATERIALES Ing.Tatiana .Dranichnikova	CALCULO DIFERENCIAL Mat. Luis Felipe Ortega	TOPOGRAFIA Ing.Luis Barros	11H:00 13H:00
15H:30 A 17H:30	QUÍMICA DE MATERIALES Ing.Tatiana .Dranichnikova	DIBUJO Arq.Doris Andrade	TÉCNICAS DE EXPRESIÓN Lcda.Enith Marcillo	DIBUJO Arq.Doris Andrade	ESTATICA Ing.Fernando Ulloa		13H:00 15H:00

Fuente: Director de la carrera de ingeniería civil

Esta forma de generar horarios es la más común tomando en cuenta que existen variaciones, que permiten el manejo de horarios a cada uno de los directores.

1.4.1.3. Generación en hoja de cálculo con color

Está forma de generar horarios es la más amigable que se puede observar dentro de las distintas maneras que los directores de carrera poseen para generar sus horarios, ya que permite de manera visual identificar cruces.

De base se tiene una tabla con un horario modificado, en el que se colocan como columnas los niveles con sus grupos y como filas los días con las horas respectivas, mostrada en la figura 6.

Figura 6: Imagen de formato para horario de la carrera de ingeniería eléctrica

DIA	HORAS	PREUNIVERSITARIO 1		PREUNIVERSITARIO 2		PRIMER NIVEL G1		PRIMER NIVEL G2		SEGUNDO NIVEL G1	
		Cátedra	Docente	Cátedra	Docente	Cátedra	Docente	Cátedra	Docente	Cátedra	Docente
LUNES	15:00-16:00										
	16:00-17:00										
	17:00-18:00										
	18:00-19:00										
	19:00-20:00										
	20:00-21:00										
MARTES	15:00-16:00										
	16:00-17:00										
	17:00-18:00										
	18:00-19:00										
	19:00-20:00										
	20:00-21:00										

Fuente: Director de la carrera de ingeniería eléctrica

Luego se procede a colocar las materias y profesores, identificando a cada profesor con un color único, de tal forma que, en una fila no se pueda observar colores duplicados, esto lo puede observar en la figura 7.

Figura 7: Imagen de formato para horario de la carrera de ingeniería eléctrica

DIA	HORAS	PREUNIVERSITARIO 1		PREUNIVERSITARIO 2		PRIMER
		Cátedra	Docente	Cátedra	Docente	Cátedra
LUNES	15:00-16:00	Química Bás	S. Olmedo			INGLÉS I
	16:00-17:00	Química Bás	S. Olmedo			
	17:00-18:00	Matemáticas	M. Pazmiño			Algebra Lineal
	18:00-19:00	Matemáticas	M. Pazmiño			Algebra Lineal
	19:00-20:00	Física	D. Trujillo	Expresión	W.Freire	Antropología Cristiana
	20:00-21:00	Física	D. Trujillo	Expresión	W.Freire	Antropología Cristiana
MARTES	15:00-16:00					INGLÉS I
	16:00-17:00					
	17:00-18:00	Expresión	W.Freire	Matemáticas	X. Espinoza	Algebra Lineal
	18:00-19:00	Expresión	W.Freire	Matemáticas	X. Espinoza	Algebra Lineal
	19:00-20:00	Matemáticas	M. Pazmiño	Expresión	W.Freire	Cálculo Diferencial
	20:00-21:00	Matemáticas	M. Pazmiño	Expresión	W.Freire	Cálculo Diferencial
COLES	15:00-16:00	Matemáticas	M. Pazmiño	Matemáticas	X. Espinoza	INGLÉS I
	16:00-17:00	Matemáticas	M. Pazmiño	Matemáticas	X. Espinoza	
	17:00-18:00			Física	M. Pazmiño	Cálculo Diferencial

Fuente: Director de la carrera de ingeniería eléctrica

1.4.1.4. Validación de cruces entre carreras

El proceso de validación de cruce entre carreras no existe, ya que los cruces se llegan a conocer cuando se ingresa los horarios generados por los directores en el sistema "SNA" (Universidad Politécnica Salesiana, 2009) de la Universidad Politécnica Salesiana. En esta unificación aparecen inconsistencias. Para resolver esta inconsistencia los directores de carrera negocian en conjunto con el profesor afectado, para encontrar una solución y así poder el sistema "SNA" permita el ingreso de los horarios sin errores.

1.4.1.5. Publicación de horarios

Una vez resueltos todos los tipos de cruce e ingresados al sistema SNA, los directores y profesores pueden consultar e imprimir los horarios. Adicional a esto, son cargados en la página Web de la Universidad Politécnica Salesiana.

1.4.1.6. Generación de cambios

Esto se da debido a cambios por inconvenientes de última hora o no contemplados dentro de la elaboración de horarios, por ejemplo:

- Salida de un profesor.
- Cambio de profesor.

- Apertura de un nuevo grupo.
- Cambios de disponibilidad.

Los cuales son analizados y de ser factible se realizan los procesos mencionados anteriormente hasta obtener un nuevo horario.

1.5. OMT (Object Modeling Technique)

El análisis y diseño orientado a objetos, es una forma de plasmar mediante modelos cualquier tipo de problemática como: sistemas, empresas, información, entre otros; que permite a expertos o no, visualizar el problema y el proceso para solucionarlo de forma clara y concisa, permitiendo detectar rápidamente omisiones en el análisis o diseño e incorporar nuevos actores y que estos entiendan eficazmente el proceso en que se está trabajando. Permitiendo a los actores comprender de forma abstracta antes de pensar en la solución definitiva.

A nivel de sistema de software permite organizar conceptos en el dominio de la aplicación y no la visualización de la programación. Lo cual permite optimizar los recursos, obtener un mejor resultado y reutilizar los objetos en nuevas aplicaciones o modificarlos en la misma hasta obtener el sistema final.

Partiendo de esto, la metodología OMT es la que más se apega a las necesidades actuales de desarrollo de software.

El desarrollo de esta metodología identifica claramente cuatro fases que son:

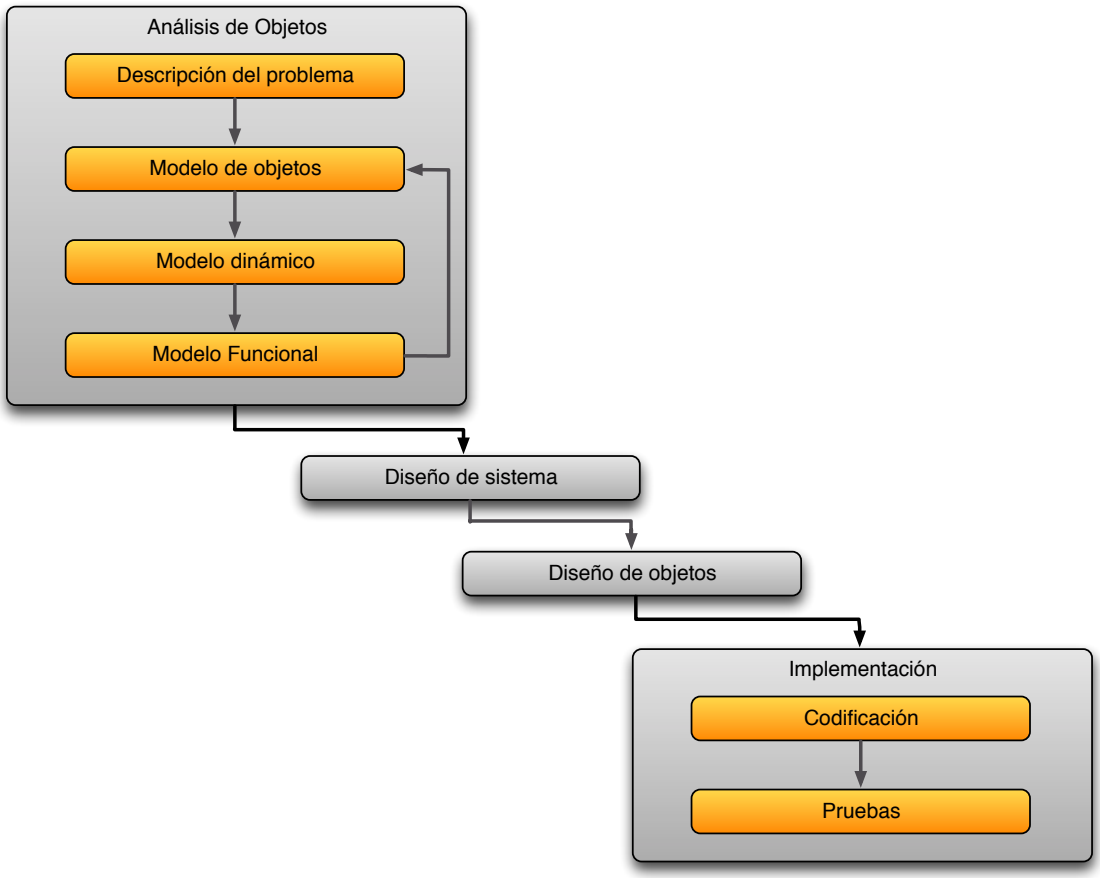
Análisis de objetos: se basa en entender y modelar gráficamente el problema desde la perspectiva de la funcionalidad que tendrá la aplicación.

Diseño del sistema: se realiza el análisis de la arquitectura del sistema. Dividiendo al sistema en subsiste, sin perder el enfoque inicial.

Diseño de objetos: se basa en el análisis de objetos y lo optimiza, a la vez que incorpora características más a detalle; buscando una definición final para implementarlo.

Implementación: es el proceso de programación, basados en los principios de la ingeniera de software, buscando principalmente flexibilidad. En conjunto se realizan pruebas para validar el correcto funcionamiento.

Figura 8: Ciclo de vida de OMT



Fuente: (Rumbaugh, 1996, p. 3)

A pesar que la Figura 8 describe a esta técnica como lineal, en realidad el proceso es iterativo durante el desarrollo. Repitiendo los pasos para afinar los detalles y añadir características o clarificar las mismas sin afectar los avances, pero sobretodo evitar que se generen errores e incongruencias.

1.5.1. Análisis de objetos

El análisis identifica de forma clara el problema y lo modela en objetos con una definición de estructura, clases y relaciones. Adicionalmente, se genera un diccionario de datos que explique a cada modelo.

La estructura para generar el análisis de objetos es el siguiente:

1.5.1.1. Definición del problema

Es la etapa más compleja del sistema, ya que se debe identificar de forma clara la falla que posee el proceso o sistema que se está buscando solucionar. Explotando las oportunidades de mejora para que la futura solución satisfaga las expectativas.

Por lo general, son los funcionarios de alto nivel quienes definen la situación del problema y en ocasiones hasta la posible solución; por lo que se recomienda indagar el problema desde la fuente, es decir, revisar directamente con los involucrados el proceso para detectar las falencias. Ya que una de las principales fallas del desarrollo de sistemas es la omisión o desconocimiento de proceso.

El problema debe quedar documentado de tal forma que no quede dudas, definiendo rendimiento, funcionalidad, contexto, entre otros.

1.5.1.2. Modelo de objetos

Para construir el modelo de objetos se debe seguir los siguientes pasos:

a) Identificar objetos y/o clases.

b) Crear diccionario de datos.

c) Identificar asociaciones y agregaciones entre los objetos.

d) Identificar atributos y enlaces.

e) Organizar y simplificar clases mediante herencia.

f) Verificar canales para probables consultas.

g) Realizar las iteraciones necesarias para el refinamiento del modelo.

h) Agrupar las clases en módulos.

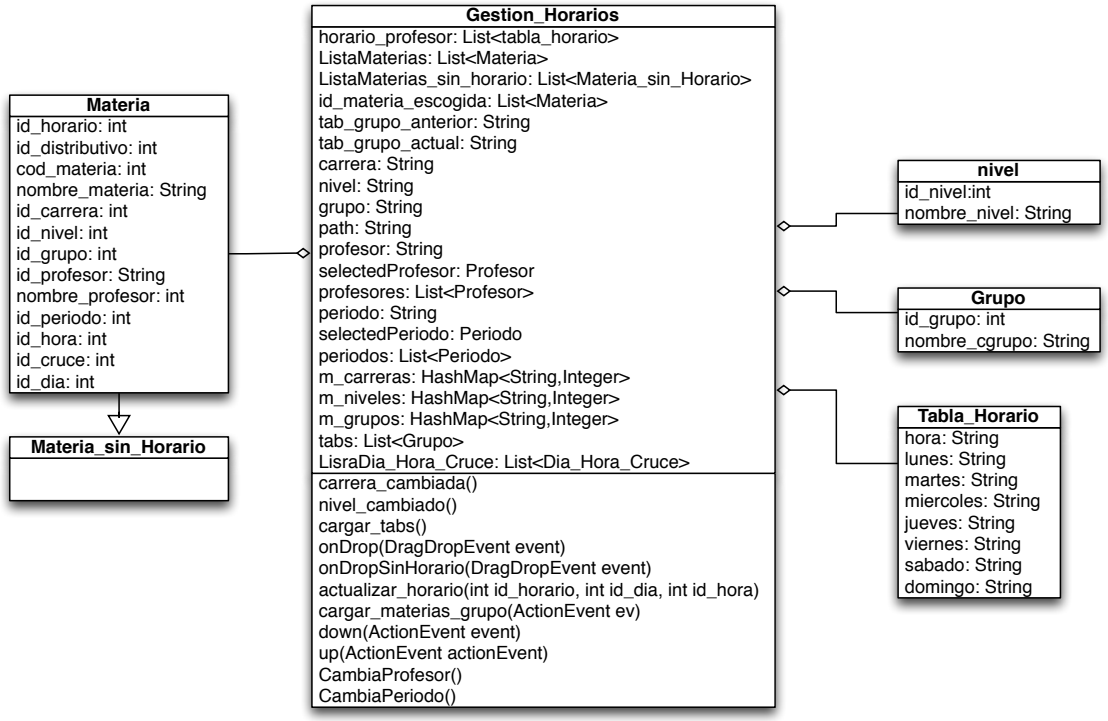
El “Modelo de objetos” se puede definir como el “Diagrama de modelo de objetos” en conjunto con el “Diccionario de datos”.

A continuación se describe como generar estos modelos:

Modelo de clases

Aquí se describen las clases encontradas en el análisis del sistema desde una visualización enfocada al problema. Identificando atributos, relaciones con las otras clases y la importancia de cada una dentro del sistema.

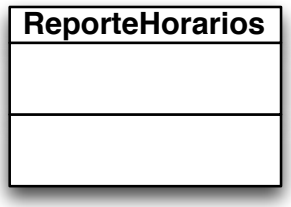
Figura 9: Modelo de clases



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Notación: Una clase se representa mediante un rectángulo, compuesto por tres separaciones, en la primer parte se coloca el nombre de la clase, en la segunda y tercera parte se pueden agregar los atributos y las operaciones, pero si no se desea agregar ninguno de ellos, es porque no son tan importantes para la comprensión del sistema, entonces el rectángulo solo se queda con el nombre de la clase.

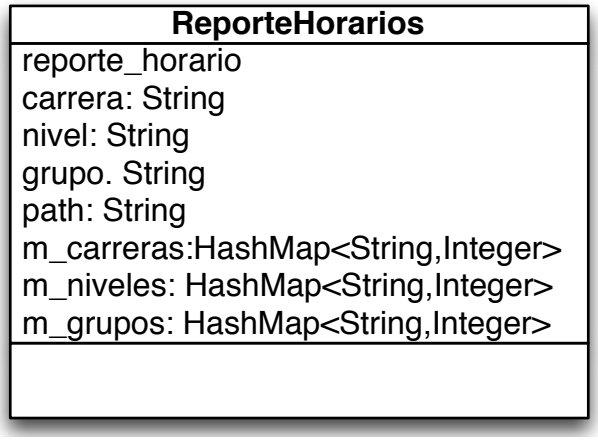
Figura 10: Representación de una clase



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Atributos: Es un dato que distingue a una clase y que puede almacenar valores para el mismo en cada instancia que genere la clase. Debe tener un nombre y el tipo de dato que va a recibir. Los atributos se colocan en la segunda parte del rectángulo de la clase, primero se coloca el nombre del atributo, después precedido de dos puntos (:) el tipo de dato que recibirá y en algunos casos se podrá especificar el valor inicial que recibe, precedido por un signo de igual (=).

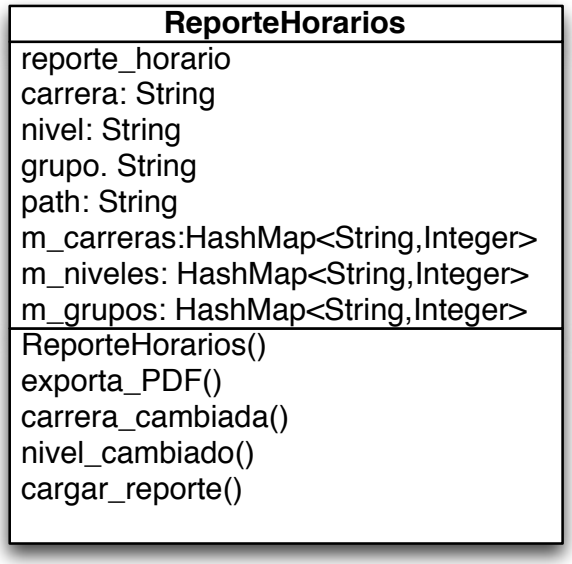
Figura 11: Representación de una clase con atributos



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Operaciones: Las operaciones son funciones que pueden realizar las instancias de una clase. Mediante ellas se pueden visualizar cuales son las responsabilidades de cada clase dentro del sistema. Se colocan en la tercera parte del rectángulo de la clase y debe contener el nombre de la operación, que puede ir seguida de una lista de argumentos entre paréntesis y del tipo de dato que regresará precedido de dos puntos (:).

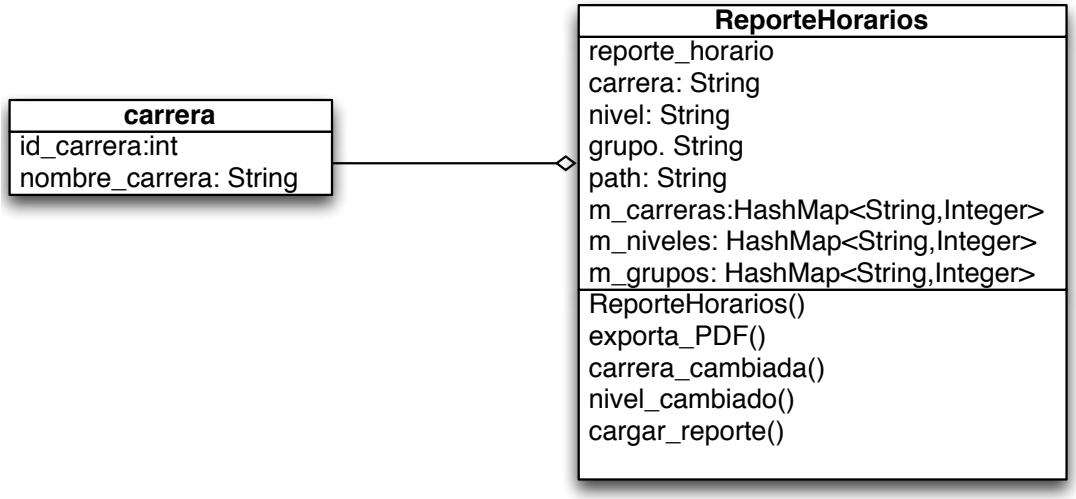
Figura 12: Representación de una clase con atributos y operaciones



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Relaciones: Representan los enlaces entre las instancias dentro del diagrama. Se representan mediante una línea que conecta a las instancias junto con el nombre de la asociación, por lo general es un verbo.

Figura 13: Representación de relación entre clases



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Simbología: Se representa las relaciones entre clases mediante líneas continuas en base a la tabla 1.

Tabla 1: Tabla de simbología de relaciones entre clases

Nombre	Símbolo	Descripción
Asociación		Asociación bidireccional entre clases con roles y multiplicidad
Agregación		Asociación entre los componentes y la clase que la compone
Composición		Relación es parte de
Dependencia		Relación de dependencia
Generalización		Relación entre una superclase que hereda sus características (atributos y operaciones) y subclases que harán suya dichas características

Fuente: (Rumbaugh, 1996, p. 7)

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

1.5.1.3. Modelo dinámico

Es la representación gráfica mediante un diagrama de estado, donde se representan los estados temporales y el manejo de los controles de la aplicación de forma secuencial en el tiempo. Para la elaboración del modelo dinámico es necesario seguir los siguientes pasos:

- Preparación de escenarios de secuencias típicas de iteración.
- Identificación de sucesos que actúan entre objetos.
- Preparar un seguimiento de sucesos para cada escenario.
- Construcción de un diagrama de estado para cada objeto.
- Comparación de los sucesos intercambiados entre objetos para verificar la congruencia.

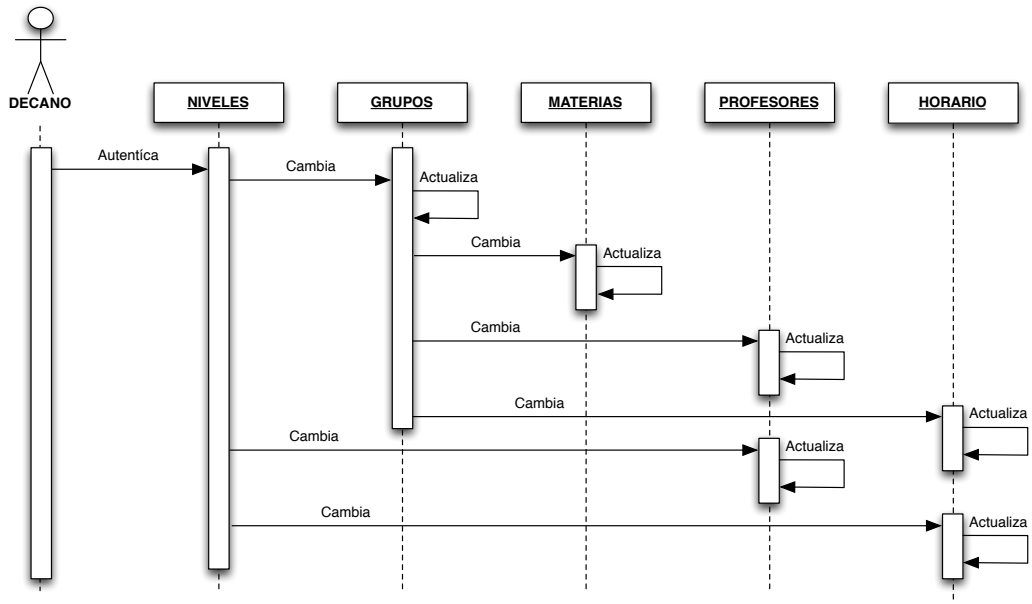
Una vez realizados los pasos anteriores, es necesario formalizar el modelo dinámico, el cual está conformado por el diagrama de flujo de sucesos y el diagrama de estados.

Diagrama de flujo de sucesos

Es la representación gráfica de los eventos que se denotan entre diferentes objetos o actores, los cuales son colocados en forma de columna con una línea vertical debajo. De este modo se puede identificar los mensajes entre los objetos y la aparición de los objetos en el tiempo.

Suceso: Es un evento que ocurre en el tiempo durante la ejecución del sistema, mediante el cual los objetos intercambian valores. Los sucesos están representados por una flecha que va desde el objeto emisor hasta el objeto receptor en forma de solicitudes que permiten la interacción entre ellos.

Figura 14: Diagrama de flujo de sucesos.



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Este diagrama será el escenario base para poder elaborar el siguiente diagrama, que es, el diagrama de estado.

Diagrama de estado

El diagrama de estados muestra de forma gráfica los estados que afectan o modifican a un objeto luego de realizar un proceso o realizar un evento, estos eventos pueden ser mensajes, restricciones, condiciones, acciones, actividades, entre otros. Y la forma en que el objeto responderá a los eventos que lo intervengan, de tal forma que sea posible identificar de forma secuencial los caminos que tomará cada objeto.

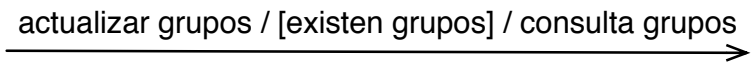
Estado: Los estados se representan generalmente por rectángulos redondeados (identificado con un nombre que lo represente) y definen la respuesta que tendrá el objeto luego de ser afectado por una acción o suceso, en determinado tiempo dentro de la ejecución del sistema.

Figura 15: Representación de estado

Fuente: (Rumbaugh, 1996, pp. 9-14)

Transiciones: Son flechas que salen desde un estado llamado estado receptor hacia otro estado llamado estado destino y representan los sucesos que afectaron al dicho estado. Cada transición es única ya que corresponde a un suceso distinto.

Figura 16: Representación de transición

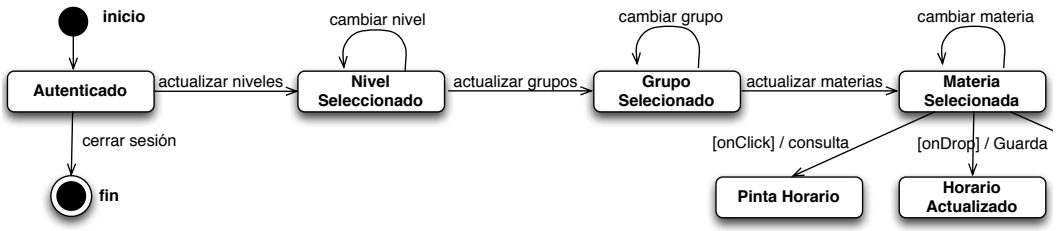


Fuente: (Rumbaugh, 1996, pp. 9-14)

Condiciones: Es una regla que evita que una transición se realice, sí dicha regla no se cumple puede volver a el estado receptor o puede cambiar a un estado destino que cumpla con dicha regla mediante otra transición.

Acción: Es una operación que se debe realizar luego de un suceso, las acciones se denotan con el nombre de la transición luego un slash "/" y el nombre de la acción que se quiere representar.

Figura 17: Diagrama de estado



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

1.5.1.4. Modelo funcional

Es la descripción gráfica de las funcionalidades que tendrá el sistema en base a la observación de las operaciones que se realizan en los objetos, enfocado a la perspectiva del problema. Sin profundizar en cuando se realizan estas operaciones (le corresponde al modelo dinámico), ni especificando que operaciones se realizan (le corresponde al modelo de objetos).

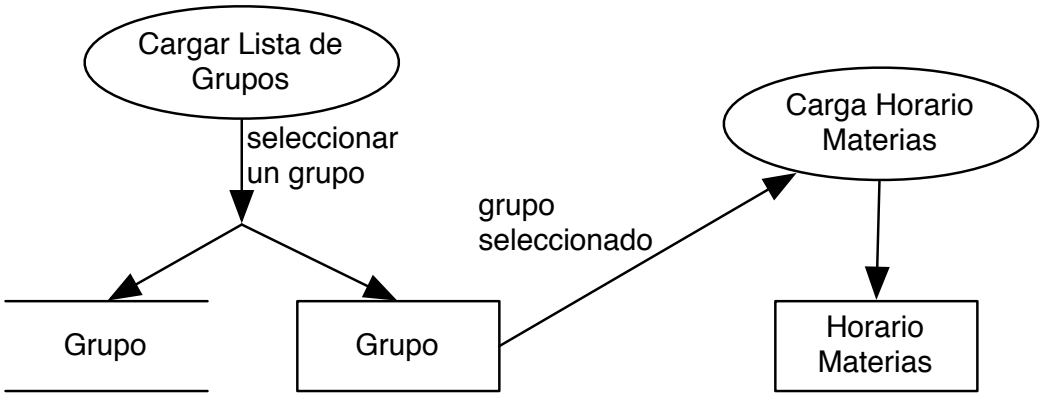
Tras lo mencionado, se puede decir que es básicamente un diagrama de flujo de datos, donde se especifican principalmente entradas, salidas y procesos que modifican dichos valores.

Para poder construir este modelo se debe seguir los siguientes pasos:

- Identificar los parámetros de entradas y salidas.
- Identificar las dependencias funcionales, mediante diagrama de flujos de datos.
- Describir las funciones existentes.
- Identificar las restricciones del sistema.
- Detallar los criterios de optimización.

La documentación formal de este diagrama está compuesto por el diagrama de flujo de datos y las restricciones.

Figura 18: Diagrama de flujo de datos



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Para entender este modelo es necesario conocer los siguientes términos y como se representan:

Proceso: Es la aplicación de una operación a una clase, los procesos manejan las entradas y se encargan de transformar esos datos en unos nuevos, con la lógica que la operación defina. Se representa con una elipse.

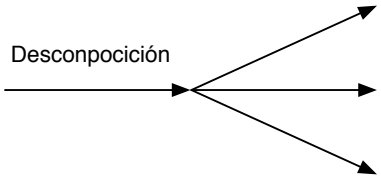
Figura 19: Representación gráfica de proceso

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Flujo de datos: Es la definición del camino que los datos deberán tomar y se encarga de conectar la salida de un objeto o proceso con la entrada de otro objeto o proceso. Se representa en el diagrama con una flecha, la cual puede llevar el nombre, valor o tipo de dato.

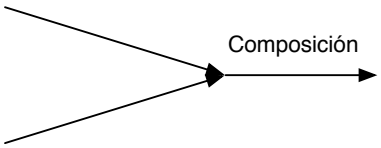
Adicional al transporte de datos permite duplicar o copiar, componer o descomponer valores.

Figura 20: Representación gráfica de la descomposición de valores, diagrama de flujo de datos



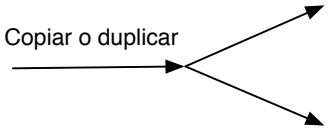
Fuente: (Rumbaugh, 1996, pp. 9-14)

Figura 21: Representación gráfica de la composición de valores, diagrama de flujo de datos



Fuente: (Rumbaugh, 1996, pp. 9-14)

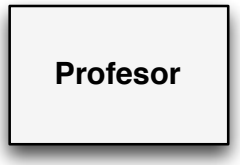
Figura 22: Representación gráfica de la copia de valores, diagrama de flujo de datos



Fuente: (Rumbaugh, 1996, pp. 9-14)

Actor: Son objetos autónomos que pueden generar sus propios datos o consumir los mismos. También se los llama terminadores, ya que actúan como el fin del flujo de datos. Se lo representa mediante un rectángulo y lleva el nombre del objeto dentro de dicho rectángulo.

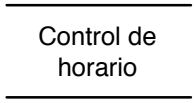
Figura 23: Representación gráfica de actor, diagrama de flujo de datos



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Almacén de datos: Es un objeto que permite almacenar datos o el acceso a los mismos, gráficamente se representa con dos barras paralelas y el nombre del almacén en el centro.

Figura 24: Representación gráfica de almacén de datos, diagrama de flujo de datos



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Definido los tres modelos, se realiza la iteración de manera global, para ello se debe:

- Comparar los tres modelos con la definición del problema y con el conocimiento del dominio de la aplicación.
- Agregar operaciones que se describe en el modelo funcional.
- Validar clases, atributos, asociaciones y operaciones de tal manera que resulten congruentes
- Generar escenarios para validar los modelos, tratando que los escenarios sean cada vez más detallados.
- Se repite el proceso con los pasos anteriores, hasta que el análisis satisfaga la definición de problema.

Una vez realizado todo el proceso, es necesario formalizar el análisis en un documento que se denomina documento de análisis, el cual contiene la definición del problema, el modelo de objetos, el modelo dinámico y el modelo funcional.

Este documento contiene el análisis, diseño e implementación de forma gráfica tomando como base los modelos en los cuales se ha venido trabajando.

1.5.2. Diseño del sistema

Es la definición que se tomará para resolver el problema y a la vez para construir la solución que se busca, tomando decisiones sobre la organización del sistema en subsiste y definiciones de optimización tanto a nivel de hardware, como software. Por ejemplo, si un subsistema debe estar asignado a uno o varios procesadores, para un mejor rendimiento de la aplicación.

La metodología define claramente los siguientes 8 pasos (Rumbaugh, 1996, pp. 1-15):

1. Organizar el sistema en subsistemas.
2. Identificar la concurrencia inherente en el problema.
3. Asignar los subsistemas a procesadores y a tareas.
4. Seleccionar la estrategia básica de implementación de los almacenes de datos, en términos de estructuras de datos, archivos y bases de datos.
5. Identificar los recursos globales y determinar los mecanismos para controlar el acceso a tales recursos.
6. Seleccionar una aproximación para implementar el control del software.
7. Consideraciones de condiciones de contorno.
8. Establecimiento de prioridades de compensación.

1.5.3. Diseño de objetos

En esta actividad se toman las decisiones para la construcción del sistema, pero sin enfocarse en el lenguaje de programación a usar, herramientas de desarrollo, sistema de base de datos, características de hardware, entre otros. En sí, el diseño de objetos es llevar el problema de la vida real a una interpretación gráfica con objetos, con sus características y propiedades, uno de los autores de esta metodología James Rumbaugh, sugiere las siguientes etapas (Rumbaugh, 1996, pp. 1-15):

1. Obtención de las operaciones para el modelo de objetos a partir de los demás modelos.
2. Diseño de algoritmos para la implementación de las operaciones.
3. Optimización de las vías de acceso a los datos.
4. Implementar el control del software completando la aproximación seleccionada durante el diseño del sistema.
5. Ajuste de la estructura de clases para incrementar la herencia.
6. Diseño de la implementación de las asociaciones.

7. Se determina la representación exacta de los atributos que son objetos.

8. Empaquetamiento de las clases y asociaciones en módulos.

1.5.4. Implementación del sistema

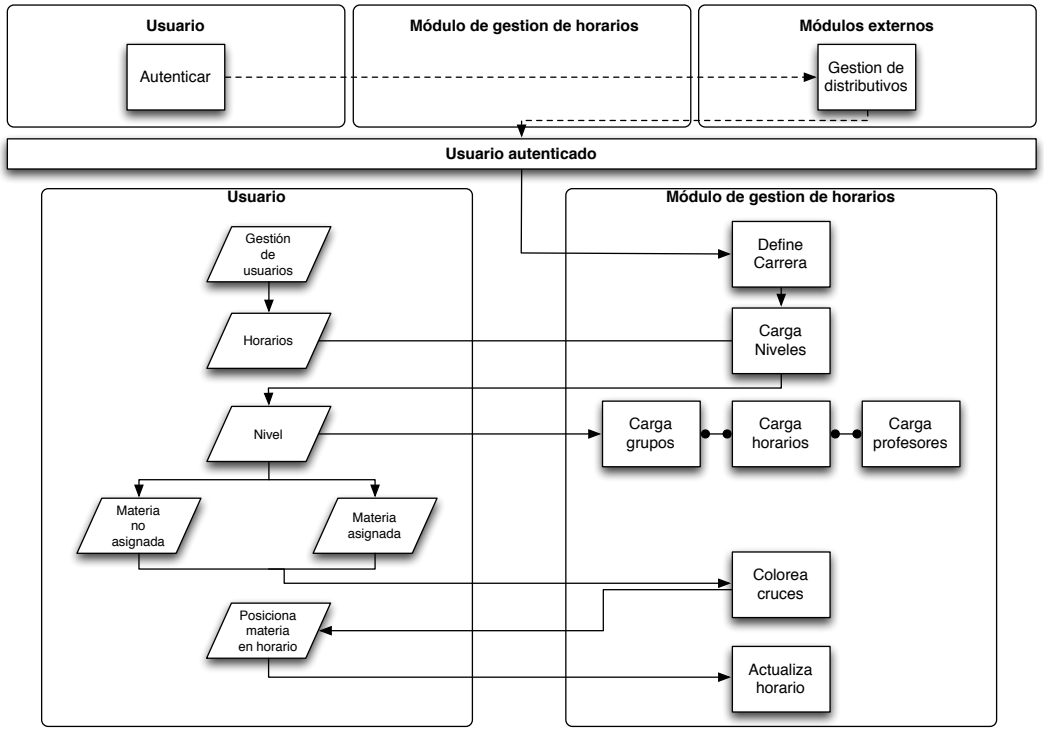
Es la etapa en donde se codificará el análisis ya realizado, no debería ser más que la transcripción de los modelos antes desarrollados a un lenguaje de programación. Adicional a la codificación, se realizará las pruebas necesarias para validar que las decisiones de diseño que se ha tomado funcionan correctamente.

CAPÍTULO 2
ANÁLISIS Y DISEÑO

2.1. Proceso de gestión de horarios

Tras la investigación realizada sobre la generación de horarios actual, se redefinieron y automatizaron varios procesos. Con el análisis se ha logrado definir un nuevo proceso de generación de horarios, el cual será plasmado en el módulo de generación de horarios y se muestra a continuación en la figura 25:

Figura 25: Nuevo proceso de generación de horarios



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

El módulo de generación de horarios permite al director de carrera de forma muy sencilla manipular sus horarios, con alertas en tiempo real que le indicarán:

Cruce por disponibilidad: esta alerta indica al director de carrera, dentro del horario, las horas y días en las que un profesor no tiene disponibilidad para dictar cátedra.

Cruce interno: esta alerta indica al director de carrera, dentro del horario, las horas y días que el profesor ya tiene asignadas materias dentro la misma carrera.

Cruce externo: esta alerta indica al director de carrera, dentro del horario, las horas y días que el profesor tiene asignadas materias, pero en una carrera distinta a la asignada al director.

Es necesario mencionar que el módulo de gestión de horarios, depende directamente de los módulos de gestión de distributivos, gestión de docentes y administración de carreras, ya que deben llenar la base de datos con la información necesaria para que el módulo de gestión de horarios funcione correctamente.

2.2. Análisis de Viabilidad

2.2.1. Análisis económico

Durante el proceso de desarrollo del módulo de gestión de horarios, se realizarán varios gastos, que serán asumidos por los tesistas. El detalle de gastos se describe en la tabla 2, que se muestra a continuación.

Tabla 2: Tabla de detalle de gastos del proyecto

Módulo de gestión de horarios				
Ítem	**Coste c/u**	**Cant.**	**Meses**	**Coste total**
Servicios básicos	20	1	9	180
Internet	35	1	9	315
Transporte	2	10	9	180
Telefonía móvil	1	20	9	180
Útiles de oficina	10	4	9	360
Libros	70	3	1	210
Ordenador	800	2	1	1600
Impresor	70	1	1	70
Hora hombre	5	100	9	4500
Derechos de grado	485	2	1	970
TOTAL				**8565**

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

2.2.2. Análisis técnico

En base a la preparación recibida y experiencia por parte de los desarrolladores del módulo de generación de horarios, la factibilidad de desarrollo es óptima, ya que se hará uso de herramientas conocidas.

A nivel de implementación del módulo de gestión de horarios, se puede observar que no existe mayor complicación, puesto que se hará uso de la infraestructura tecnológica que la Universidad Politécnica Salesiana posee y con herramientas de software libre. Todas estas herramientas poseen información para la instalación, implementación y mantenimiento.

2.2.3. Análisis legal

Para evitar el pago de licencias se usarán herramientas con licencia GNU, que permite el uso libre e ilimitado para desarrollo, implementación y puesta en marcha del módulo de gestión de horarios.

2.2.4. Análisis operativo

El principal aporte del módulo de generación de horarios es la unificación de procesos para la creación, modificación, sociabilización de horarios y la solución de cruces entre carreras.

2.2.5. Conclusión

Tomando en cuenta el análisis económico, técnico, legal y operativo se determina que el desarrollo de este proyecto es factible, ya que permite unificar el proceso de generación de horarios, optimiza notablemente el proceso actual y brinda un gran apoyo a la tarea de generar horarios.

Las herramientas actuales permiten el desarrollo del módulo de generación de horarios con funcionalidades muy amigables, el uso de herramientas con licencias GNU reducen notablemente el costo del sistema. Mientras que el costo del desarrollo será asumido por los tesistas.

Se debe tomar en cuenta que todo sistema es susceptible a la adaptación por parte del usuario; en el caso de la Universidad Politécnica Salesiana existen directores que aún se apegan a procesos manuales.

2.3. Documento de requerimientos

El presente documento contiene las directrices necesarias para la elaboración del módulo de gestión de horarios y las necesidades de los directores de carrera al momento de realizar sus horarios.

2.3.1. Propósito

El propósito de este documento es recolectar de forma detallada las necesidades de los directores de carrera, base para el desarrollo del módulo de gestión de horarios.

Definir las interfaces necesarias y su funcionalidad, las operaciones y reglas fundamentales para la elaboración de horarios.

Servir de guía para los tesistas, que serán los encargados de diseñar, desarrollar y probar el módulo de gestión de horarios.

2.3.2. Alcance

El módulo de gestión de horarios será diseñado para satisfacer las necesidades de los directores de carrera al momento de realizar horarios, dentro de una interfaz ágil y sencilla.

Buscando la centralización de procesos y la optimización de recursos: humano, técnico, laboral. El módulo de gestión de horarios será parte del sistema "UPS Schedule", mismo que será integrado e implementado dentro de la infraestructura tecnológica de la Universidad Politécnica Salesiana.

2.3.3. Definiciones, acrónimos y abreviaturas

UPS: Abreviatura de Universidad Politécnica Salesiana.

Carrera: Hace referencia a la oferta académica de la Universidad Politécnica Salesiana en cuanto a opciones de titulación, por ejemplo Ingeniería de Sistemas, Administración de Empresas, Ingeniería Ambiental, entre otros.

Nivel: Es la división curricular de las carreras que, por lo general tienen una duración de seis meses (un semestre). Sirven para separar de manera estructurada a la carrera y validar progresivamente el conocimiento que el estudiante va adquiriendo con el transcurso del tiempo.

Grupo: Es una forma de organización del alumnado (paralelo) que se encuentra cursando el mismo nivel, sirve principalmente para dividir en números manejables de estudiantes.

Materia: Es la instrucción formativa o transferencia de conocimiento, impartida por los docentes de la Universidad Politécnica Salesiana en de cada nivel a los estudiantes, consiste en el estudio de determinada área del conocimiento.

Hora de clase: Es la unidad de tiempo mínima de instrucción recibida por los estudiantes, la hora de clase hace referencia a la característica de una materia, es igual a la unidad de tiempo hora (60 minutos) y define el inicio de instrucción.

2.3.4. Referencias

Este documento usa como base los archivos para la generación de horarios de los diferentes directores de carrera enmarcados dentro las recomendaciones de:

IEEE, IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society, 1998.

2.3.5. Información general

A continuación se define los detalles sobre el módulo de gestión de horarios, los requerimientos que se han logrado definir durante la investigación y detalle de las funcionalidades y limitaciones.

2.3.6. Descripción general

2.3.6.1. Perspectivas del producto

El módulo de generación de horarios forma parte del sistema "UPS Schedule", este módulo será desarrollado como una aplicación Web, que se integrará en el futuro al sistema. Al ser parte de un sistema más complejo, los recursos como: servidor aplicaciones, base de datos serán compartidas.

A pesar de ser parte del sistema "UPS Schedule", el módulo de generación de horarios debe funcionar de forma independiente, buscando agilitar, facilitar y organizar el proceso de generación de horarios.

2.3.6.2. Funciones del producto

El módulo de gestión de horarios es de vital importancia dentro del sistema "UPS Schedule", ya que permitirá a los directores de carrera generar los horarios y modificarlos.

La característica principal del módulo es que debe usar tecnologías actuales, que permita agilidad y facilidad al usuario, para disminuir la complejidad de elaborar horarios.

2.3.6.3. Características de usuario

El módulo de generación de horarios está enfocado a un único usuario que es el director de carrera.

El director de carrera es la máxima autoridad en una carrera, es de responsabilidad de este usuario generar horarios dentro del módulo de generación de horarios.

2.3.6.4. Restricciones

El módulo de generación de horarios depende de la información de materias, distributivo y disponibilidad de profesores, la cual debe estar cargada previamente en la base de datos.

El desarrollo debe ser completamente Web, para ser alojado en la infraestructura tecnológica de la Universidad Politécnica Salesiana.

2.3.6.5. Suposiciones y dependencias

Al ser el módulo de gestión de horarios parte del sistema "UPS Schedule" debe ser compatible para su integración:

- Implementación en la plataforma GNU/Linux.
- Aplicación Web.
- Uso de PrimeFaces, tecnología JAVA.
- Uso de herramientas con licencia GNU, tanto para el diseño, desarrollo e integración.

Es necesario indicar que el desarrollo se encuentra basado en las necesidades actuales de las carreras y de existir cambios en la estructura, será necesario revisar los requerimientos.

2.3.6.6. Requisitos para futuras versiones del sistema

La principal tarea es integrar e implementar el módulo al sistema "UPS Schedule".

En cuanto a la evolución del módulo de generación de horarios se podría buscar en un futuro la sincronización de los horarios generados con el "SNA" de la Universidad Politécnica Salesiana y de igual forma poder obtener la información de materias, docentes y distributivos.

2.3.7. Requerimientos específicos

2.3.7.1. Interfaz de usuario

La interfaz de usuario debe 100% Web, intuitiva y sencilla; utilizando características del desarrollo Web moderno que permitan un manejo eficiente, rápido y amigable con el usuario.

En cuanto al aspecto visual es necesario que los colores permitan un uso del aplicativo por períodos largos, sin descanso.

2.3.7.2. Interfaz de hardware

Durante el desarrollo se buscará que las limitaciones de hardware del cliente no determiné el funcionamiento de la aplicación, para esto de se busca que el servidor de aplicaciones y base de datos sea robusto de tal forma que soporte los proceso necesarios para el correcto funcionamiento.

En base a lo mencionado, el aplicativo deberá ser funcional bajo cualquier arquitectura actual 32 bits y 64 bits. En cuanto a la interfaz de comunicación deberá ser una interfaz de red, con una velocidad en base los parámetros actuales de 100Mbps o superior.

Para sus características graficas es necesario que la tarjeta de video soporte una resolución de 1024 x 768.

2.3.7.3. Interfaz de software

La aplicación debe ser totalmente independiente de las aplicaciones que actualmente usa la Universidad Politécnica Salesiana, adicional, todo el desarrollo se debe hacer con herramientas de software libre.

2.3.7.4. Interfaz de comunicación

La comunicación cliente servidor deberá manejar un protocolo HTTP tanto desde una intranet como desde internet y TCP/IP entre servidores.

2.3.8. Requerimientos funcionales

Es necesario mencionar que el módulo de generación de horarios está enfocado al usuario director de carrera, por lo que los requisitos que a continuación se mencionan, engloban los requerimientos de este único usuario.

2.3.8.1. Requerimiento funcional 1.1: Generación de horarios.

Descripción	Es necesario poder crear un horario, haciendo uso de las materias, niveles y profesores para un período especifico
Precondición	Profesores, carreras, niveles, grupos, materias, distributivos, disponibilidad, días, horas, tipos de cruce se encuentren precargados en la base de datos. Usuario autenticado y valores en carrera, nivel y grupo
Entrada	Selección de materia no asignada
Proceso	La materia es validada con el distributivo y disponibilidad del profesor, también con otros horarios y en base a eso se define un tipo de cruce
Salida	Tipo de cruce

2.3.8.2. Requerimiento funcional 1.2: Modificación de horarios.

Descripción	Las materias ya asignadas a un horario deben permitir modificarse de hora y día asignado o no asignar a un horario
Precondición	Materia asignada a un horario
Entrada	Materia
Proceso	La materia es validada con el distributivo y disponibilidad del profesor, también con otros horarios y en base a eso se define un tipo de cruce
Salida	Tipo de cruce

2.3.8.3. Requerimiento funcional 1.3: Validación de cruce en horario.

Descripción	Se debe validar los diferentes errores que impiden que un horario sea eficiente, a los que se les denominará cruce
Precondición	Materia asignada a un horario
Entrada	Materia
Proceso	Validar cruce por disponibilidad Validar cruce interno Validar cruce con otra carrera
Salida	Tipo de cruce

2.3.8.4. Requerimiento funcional 1.4: Reportes de horario de profesores.

Descripción	Se deben generar reportes de forma visual en formato PDF
Precondición	Selección de un profesor
Entrada	Profesor
Proceso	Seleccionar el horario del profesor
Salida	Horario de un profesor especifico

2.3.8.5. Requerimiento funcional 1.5: Reporte de horario por grupo.

Descripción	Se deben generar reportes de forma visual en formato PDF
Precondición	Selección de un grupo
Entrada	Grupo
Proceso	Seleccionar el horario de un grupo
Salida	Horario de un grupo específico

2.3.9. Requerimientos de rendimiento

Es necesario que la herramienta a desarrollar sea usada por los directores de carrera de toda la Universidad Politécnica Salesiana sede Quito, que son en promedio 20 usuarios.

El uso será esporádico, ya que la herramienta generalmente se usa durante el lapso de generación de horarios, pero durante ese período la herramienta necesita ser lo más eficiente y robusta, por que estará siendo usada al máximo de su capacidad. Por lo que se debe validar una disponibilidad 24/7.

Los procesos que se ejecuten en el cliente no deben sobrepasar el 10% del uso la CPU, y 50% del uso de la CPU del servidor.

El 90% de las transacciones que se realicen deben tardar menos de 1m 30s.

2.3.10. Restricciones de diseño

Los reportes no son personalizables, solo se obtienen los reportes solicitados en los requerimientos del sistema, que son reporte de horario por profesor y por grupo.

Es necesario para el correcto funcionamiento de la aplicación que el navegador, soporte Java Script.

No es posible la importación de datos al sistema.

El correcto funcionamiento del módulo de generación de horarios depende de que la información de materias, profesores y distributivos se encuentre cargada correctamente en la base de datos.

2.3.11. Atributos del sistema

El principal atributo del módulo de generación de horarios, es la facilidad que brinda al usuario al momento de realizar horarios, ya que usa tecnologías nuevas que permiten el uso de eventos arrastrar y soltar.

El desarrollo Web del módulo de gestión de horarios hace de esta aplicación accesible desde intranet e internet, sin limitantes para los usuarios.

La velocidad de respuesta del aplicativo, supera de sobremanera a los métodos usados por los directores de carrera, durante el levantamiento de información.

2.4. Diseño del sistema

2.4.1. Análisis de objetos

2.4.1.1. Definición del problema

Tomando en cuenta los procesos actuales para la generación de horarios, se ha notado que son muy susceptibles a errores, ya que se basa en experiencia, observación y memoria temporal, generando más de un problema cuando se quiere asignar una materia al horario.

El problema más notorio es la validación de cruces, pues este requiere de un gran recurso de tiempo y de mucha concentración, porque se valida dos tipos de cruce al mismo instante.

El primero y más importante es certificar que un profesor no haya sido asignado en dos grupos o niveles a la vez, entendiéndose a la vez como mismo día y hora.

El segundo es la disponibilidad del profesor, esta se valida de forma objetiva porque no siempre se encuentra una solución, debido a que la validación se realiza únicamente de manera visual.

Este problema se da principalmente porque no tiene algún mecanismo de alerta que permita disminuir el error humano.

En ocasiones, el director de carrera cierra horarios, pasando por alto que aún quedan cruces. Hasta que los horarios son cargados al SNA; siendo este sistema su validador.

Hay también que tomar en cuenta que existen profesores que dictan clases en más de una carrera, y es aquí cuando aparece otro inconveniente, el hecho de no poder garantizar si existe o no un cruce con otra carrera, ya que los horarios no lo hacen todos los directores en conjunto. Este tipo de cruce se llega a conocer una vez que todos los directores han terminado sus horarios y ya han entregado los mismos para ser ingresados al SNA. Regresando al problema de ser revisado y notificado.

En caso de que exista un cruce con otra carrera es más complicado; porque los directores de carrera involucrados deben encontrar en consenso una nueva solución, validando todo lo mencionado anteriormente.

No se debe olvidar todo el tiempo transcurrido, lo generando mayor presión, causando estrés.

2.4.1.2. Modelo de objetos

Diccionario de datos

En la tabla 3 se observa el diccionario de datos, detallando las clases y objetos a usar.

Tabla 3: Tabla de diccionario de datos

DATO	TIPO	OBSERVACIÓN
Dia_Hora_Cruce	Sustantivo	Clase
id_dia	Sustantivo	Atributo de la clase
getId_dia	Verbo	No relevante
setId_dia	Verbo	No relevante
id_hora	Sustantivo	Atributo de la clase
getId_hora	Verbo	No relevante
setId_hora	Verbo	No relevante
tipo_cruce	Sustantivo	Atributo de la clase
getTipo_cruce	Verbo	No relevante
setTipo_cruce	Verbo	No relevante
id_cruce	Sustantivo	Atributo de la clase
getId_cruce	Verbo	No relevante

DATO	TIPO	OBSERVACIÓN
setId_cruce	Verbo	No relevante
cruce	Sustantivo	Atributo de la clase
getCruce	Verbo	No relevante
setCruce	Verbo	No relevante
color	Sustantivo	Atributo de la clase
getColor	Verbo	No relevante
setColor	Verbo	No relevante
Materia	Sustantivo	Clase
id_horario	Sustantivo	Atributo de la clase
id_distributivo	Sustantivo	Atributo de la clase
cod_materia	Sustantivo	Atributo de la clase
nombre_materia	Sustantivo	Atributo de la clase
id_carrera	Sustantivo	Atributo de la clase
id_nivel	Sustantivo	Atributo de la clase
id_grupo	Sustantivo	Atributo de la clase
id_profesor	Sustantivo	Atributo de la clase
nombre_profesor	Sustantivo	Atributo de la clase
id_período	Sustantivo	Atributo de la clase
Materia_sin_horario	Sustantivo	Clase
Profesor	Sustantivo	Clase
id_profesor	Sustantivo	Atributo de la clase
getId_profesor	Verbo	No relevante
setId_profesor	Verbo	No relevante
nombre_profesor	Sustantivo	Atributo de la clase
getNombre_profesor	Verbo	No relevante
setNombre_profesor	Verbo	No relevante
Foto	Sustantivo	Atributo de la clase
getFoto	Verbo	No relevante
setFoto	Verbo	No relevante
id_período	Sustantivo	Atributo de la clase
getId_período	Verbo	No relevante
setId_período	Verbo	No relevante
Período	Sustantivo	Clase
numero	Sustantivo	Atributo de la clase
getNumero	Verbo	No relevante
setNumero	Verbo	No relevante
descripcion	Sustantivo	Atributo de la clase

DATO	TIPO	OBSERVACIÓN
getDescripcion	Verbo	No relevante
setDescripcion	Verbo	No relevante
estado	Sustantivo	Atributo de la clase
getEstado	Verbo	No relevante
setEstado	Verbo	No relevante
Carrera	Sustantivo	Clase
id_carrera	Sustantivo	Atributo de la clase
getId_carrera	Verbo	No relevante
setId_carrera	Verbo	No relevante
nombre_carrera	Sustantivo	Atributo de la clase
getNombre_carrera	Verbo	No relevante
setNombre_carrera	Verbo	No relevante
Nivel	Sustantivo	Clase
id_nivel:int	Sustantivo	Atributo de la clase
getId_nivel	Verbo	No relevante
setId_nivel	Verbo	No relevante
nombre_nivel	Sustantivo	Atributo de la clase
getNombre_nivel	Verbo	No relevante
setNombre_nivel	Verbo	No relevante
Grupo	Sustantivo	Clase
id_grupo	Sustantivo	Atributo de la clase
getId_grupo	Verbo	No relevante
setId_grupo	Verbo	No relevante
nombre_grupo	Sustantivo	Atributo de la clase
getNombre_grupo	Verbo	No relevante
setNombre_grupo	Verbo	No relevante
Gestion_Horarios	Sustantivo	Clase
horario_profesor	Sustantivo	Atributo de la clase
getHorario_profesor	Verbo	No relevante
setHorario_profesor	Verbo	No relevante
ListaMaterias	Sustantivo	Atributo de la clase
getListaMaterias	Verbo	No relevante
setListaMaterias	Verbo	No relevante
ListaMaterias_sin_horario	Sustantivo	Atributo de la clase
getListaMaterias_sin_horario	Verbo	No relevante
setListaMaterias_sin_horario	Verbo	No relevante
id_materia_escogida	Sustantivo	Atributo de la clase

DATO	TIPO	OBSERVACIÓN
getId_materia_escogida	Verbo	No relevante
setId_materia_escogida	Verbo	No relevante
tab_grupo_anterior	Sustantivo	Atributo de la clase
getTa_grupo_anterior	Verbo	No relevante
setTa_grupo_anterior	Verbo	No relevante
tab_grupo_actual	Sustantivo	Atributo de la clase
getTa_grupo_actual	Verbo	No relevante
setTa_grupo_actual	Verbo	No relevante
nivel	Sustantivo	Atributo de la clase
getNivel	Verbo	No relevante
setNivel	Verbo	No relevante
grupo	Sustantivo	Atributo de la clase
getGrupo	Verbo	No relevante
setGrupo	Verbo	No relevante
path	Sustantivo	Atributo de la clase
getPath	Verbo	No relevante
setPath	Verbo	No relevante
profesor	Sustantivo	Atributo de la clase
getProfesor	Verbo	No relevante
setProfesor	Verbo	No relevante
selectedProfesor	Sustantivo	Atributo de la clase
getSelectedProfesor	Verbo	No relevante
setSelectedProfesor	Verbo	No relevante
profesores	Sustantivo	Atributo de la clase
getProfesores	Verbo	No relevante
setProfesores	Verbo	No relevante
período	Sustantivo	Atributo de la clase
getPeríodo	Verbo	No relevante
setPeríodo	Verbo	No relevante
selectedPeríodo	Sustantivo	Atributo de la clase
getSelectedPeríodo	Verbo	No relevante
setSelectedPeríodo	Verbo	No relevante
períodos	Sustantivo	Atributo de la clase
getPeríodos	Verbo	No relevante
setPeríodos	Verbo	No relevante
m_carreras	Sustantivo	Atributo de la clase
getM_carreras	Verbo	No relevante

DATO	TIPO	OBSERVACIÓN
setM_carreras	Verbo	No relevante
m_niveles	Sustantivo	Atributo de la clase
getM_nivel	Verbo	No relevante
setM_nivel	Verbo	No relevante
m_grupos	Sustantivo	Atributo de la clase
getM_grupos	Verbo	No relevante
setM_grupos	Verbo	No relevante
Tabs	Sustantivo	Atributo de la clase
getTabs	Verbo	No relevante
setTabs	Verbo	No relevante
LisraDia_Hora_Cruce	Sustantivo	Atributo de la clase
getLisraDia_Hora_Cruce	Verbo	No relevante
setLisraDia_Hora_Cruce	Verbo	No relevante
Tabla_Horario	Sustantivo	Clase
hora	Sustantivo	Atributo de la clase
getHora	Verbo	No relevante
setHora	Verbo	No relevante
lunes	Sustantivo	Atributo de la clase
getLunes	Verbo	No relevante
setLunes	Verbo	No relevante
martes	Sustantivo	Atributo de la clase
getMartes	Verbo	No relevante
setMartes	Verbo	No relevante
miercoles	Sustantivo	Atributo de la clase
getMiercoles	Verbo	No relevante
setMiercoles	Verbo	No relevante
jueves	Sustantivo	Atributo de la clase
getJueves	Verbo	No relevante
setJueves	Verbo	No relevante
viernes	Sustantivo	Atributo de la clase
getViernes	Verbo	No relevante
setViernes	Verbo	No relevante
sabado	Sustantivo	Atributo de la clase
getSabado	Verbo	No relevante
setSabado	Verbo	No relevante
domingo	Sustantivo	Atributo de la clase
getDomingo	Verbo	No relevante

DATO	TIPO	OBSERVACIÓN
setDomingo	Verbo	No relevante
Reporte_Horarios	Sustantivo	Clase
Reporte_horario	Sustantivo	Atributo de la clase
getReporte_horario	Verbo	No relevante
setReporte_horario	Verbo	No relevante
nivel_cambiado	Verbo	actualiza lista de grupos para nivel seleccionado
cargar_tabs	Verbo	carga tabs con los grupos existentes para un nivel
onDrop	Verbo	despinta cruces y llama a actualizar_horario()
onDropSinHorario	Verbo	despinta cruces y llama a actualizar_horario()
actualizar_horario	Verbo	actualiza la tabla horario de la base de datos
cargar_materias_grupo	Verbo	carga recuadros con las materias para grupo seleccionado
down	Verbo	consulta cruces y pinta horario
Up	Verbo	despinta horario
CambiaProfesor	Verbo	consulta horario profesor
CambiaPeríodo	Verbo	período seleccionado
ReporteHorarios	Verbo	consulta el horario de un grupo especifico
exporta_PDF	Verbo	exporta horario en formato pdf

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Modelo de clases

En base al diccionario de datos se extrae las clases más importantes detalladas en la tabla 4.

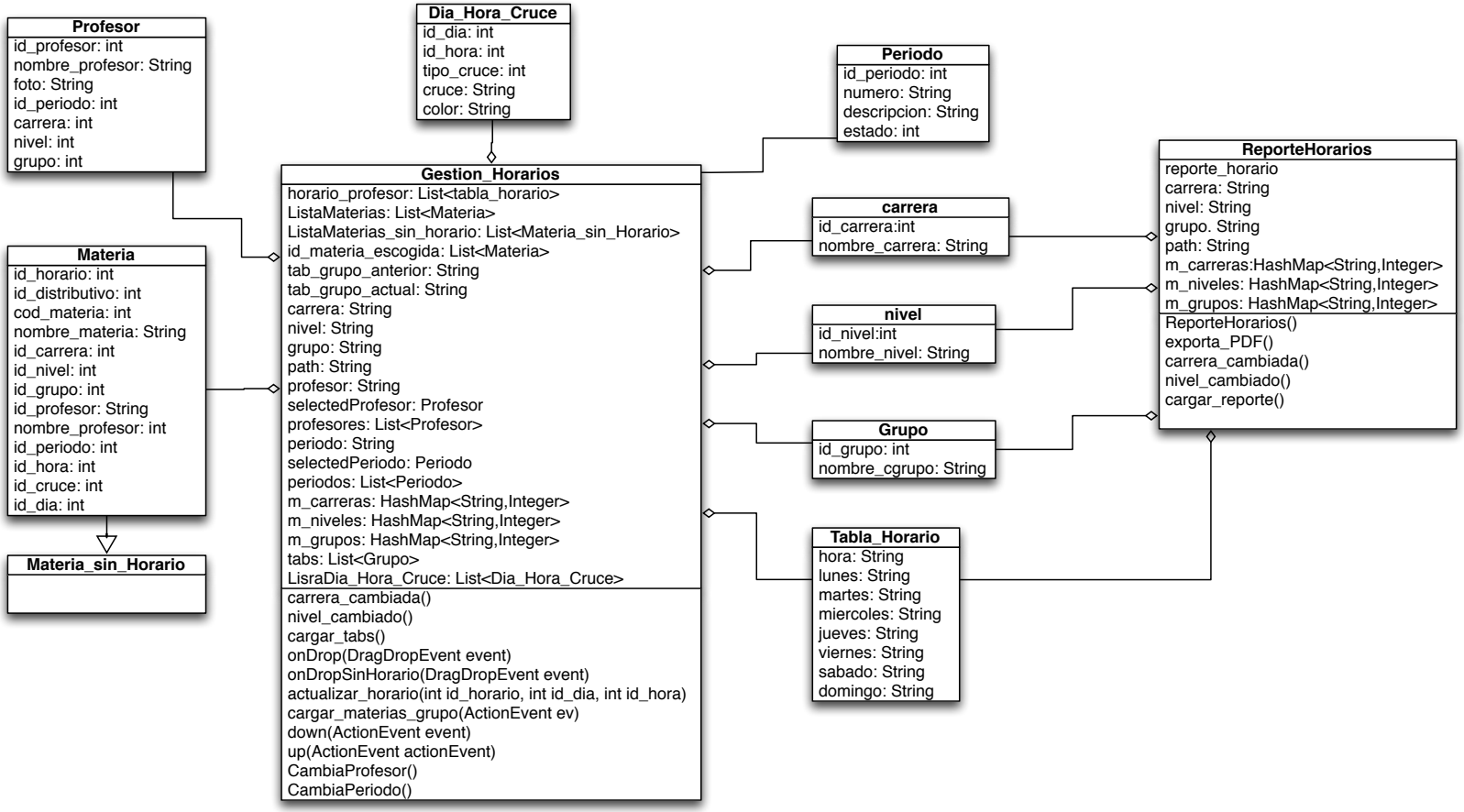
Tabla 4: Tabla de clases del módulo de gestión de horarios

Clase	Detalle
Dia_Hora_Cruce	Contiene la información con el tipo de cruce que tiene cada día y hora dependiendo de la carrera, nivel y grupo con que se esté trabajando
Materia	Contiene información de cada materia, para saber: id, carrera, nivel, grupo al que pertenece, quien la dicta, y en qué día y hora está asignado
Materia_sin_horario	Contiene información de cada materia, para saber: de carrera nivel y grupo pertenece y quien la dicta
Profesor	Contiene información de un profesor: id, nombres, apellidos, cedula o pasaporte y teléfono.
Período	Contiene información sobre cada período: id, numero de período, detalle y saber su estado para permitir su modificación
Carrera	Contiene información de la carrera: id, campus al que pertenece, nombre de la carrera y quien es el director
Nivel	Contiene información de un nivel para saber: a que carrera pertenece, id y nombre.
Grupo	Contiene información de un grupo para saber: a que carrera y nivel pertenece, id y nombre.
Gestion_Horarios	Esta clase interactúa con el navegador Web para la generación de horarios
Tabla_Horario	Contiene parte de la información de un horario; materia para cada día
Reporte_Horarios	Esta clase interactúa con el navegador Web para poder obtener reportes de horarios

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Una vez identificadas las clases más importantes, se agrega las relaciones, como se puede ver en la figura 26. Siendo las clases más importantes “Gestión Horario” y “Reporte Horario”; ya que estas son las que interactúan con la capa de presentación.

Figura 26: Modelo de clases



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

2.4.1.3. Modelo dinámico

Diagrama de secuencia

Hay tres funciones para cumplir con los objetivos del módulo de gestión de horarios, estos serán los siguientes; basados en que el director de carrera será el único usuario del sistema.

- **Gestión de horarios**

Tomando en cuenta que se quiere realizar un horario para un grupo, va a existir un filtro que será el nivel. Para poder diferenciar los grupos y administrarlos mejor; porque en casos hay grupos con el mismo nombre para todos los niveles; por ejemplo, el nivel 1 tiene grupo 1, 2 y 3; mientras que el nivel 2 también tiene el grupo 1, 2 y 3; entonces, para que se pueda diferenciar, se tiene un objeto nivel el que dirá el grupo 1, a que nivel pertenece, ya sea el nivel 1 o 2.

Cuando se quiera generar un horario; es decir manipular las materias e ir creando un horario para cada grupo, el director de carrera tiene que seguir el siguiente procedimiento:

1. Debe seleccionar un nivel de una lista de ellos (serán únicamente los niveles que corresponden a la carrera que el director pertenece), ya que no podrá ver otro objeto más.

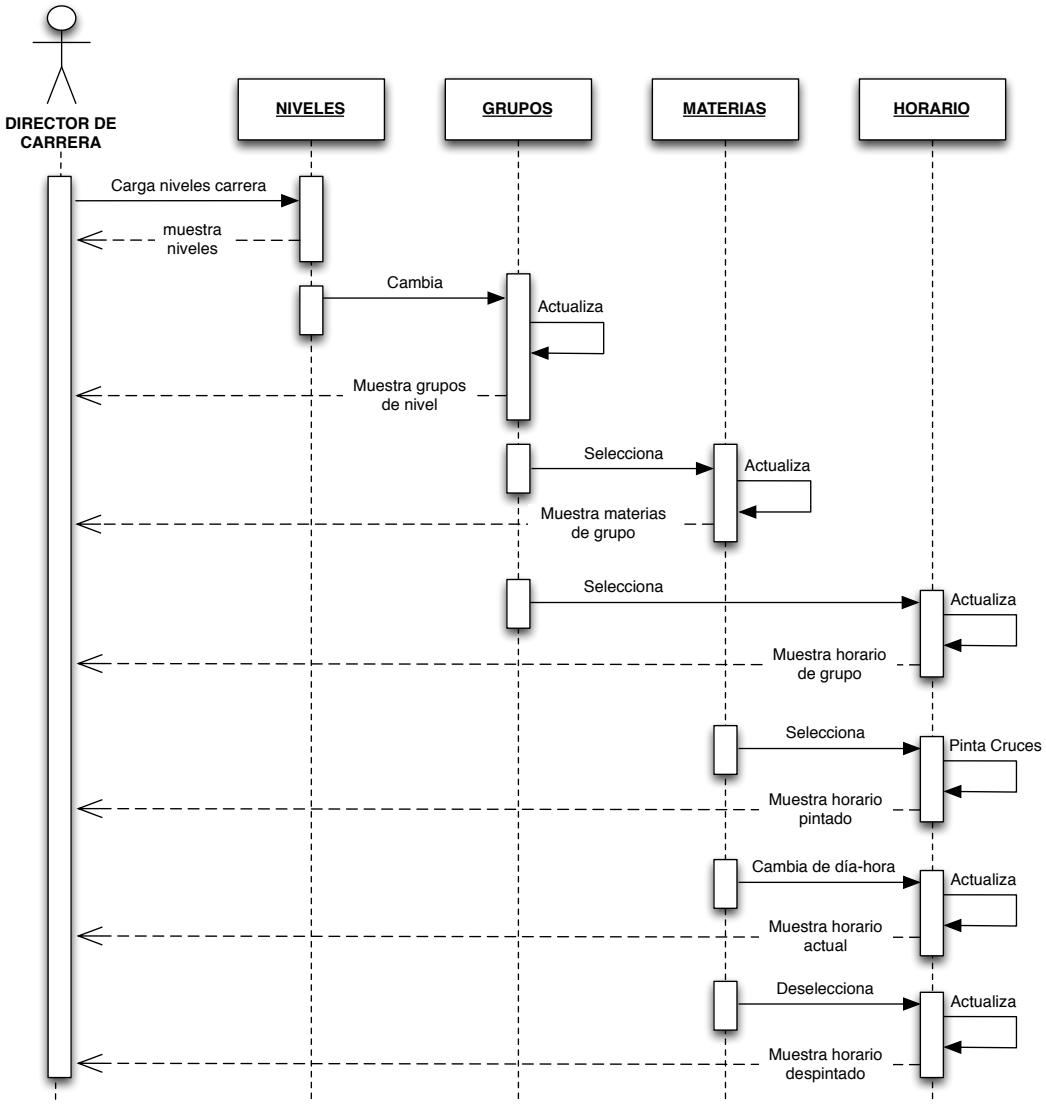
2. Debe escoger un grupo, de un listado de ellos, pertenecientes al nivel seleccionado en el paso 1, por lo que no van a aparecer grupos mientras no haya realizado el paso anterior.

3. Una vez realizado los pasos 1 y 2 se puede empezar a manipular, las materias que pertenecerán al grupo y nivel escogidos; así, para generar los horarios, se debe seleccionar la materia, arrastrar al día y hora que se lo quiere asignar. Teniendo alertas en el horario que serán pintadas, para evitar algún tipo de cruce.

Si se quiere generar el horario de otro grupo perteneciente al mismo nivel solo se debe realizar nuevamente el paso 2 y 3. Y si es perteneciente a otro nivel, entonces se debe regresar al primer paso.

Esto se explica de manera gráfica en la figura 27.

Figura 27: Flujo de sucesos, crear horarios



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

- **Consultar el horario de un profesor**.

Para el poder ver el horario de un profesor, se interactuará de la siguiente manera con los objetos principales, y estos son: niveles, grupos, materias, lista de profesores y horario del profesor.

Para esto debe seguir los siguientes pasos:

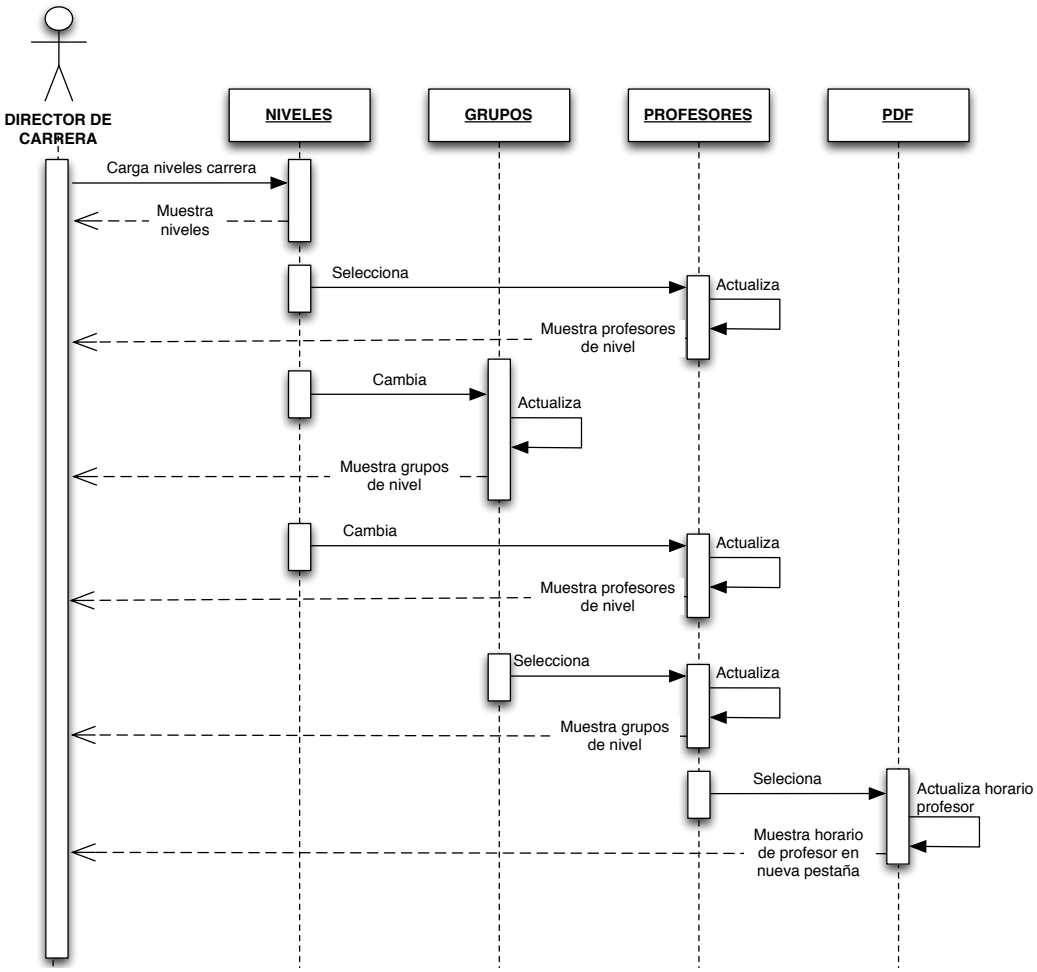
1. Seleccionar un profesor para ver su horario y si quiere filtrar la búsqueda para un profesor debe realizar los pasos siguientes.

2. Debe seleccionar un nivel de una lista de ellos (serán únicamente los niveles que corresponden a la carrera que el director pertenece), ya que no podrá ver ningún otro objeto más, exceptuando a los profesores. Y si es suficiente con ese filtro debe realizar el paso 1.

3. Debe escoger un grupo de un listado de ellos, pertenecientes al nivel seleccionado en el paso 2, por lo que no van a aparecer grupos mientras no haya realizado este paso. Este será el último filtro posible. Es decir este será el nivel más bajo de filtros. Y para ver el horario del profesor perteneciente al grupo tendrá que realizar el primer paso.

Estos filtros se han de aprovechar, ya que existen en la “Gestión de horarios”. Se puede ver lo mencionado en la figura 28.

Figura 28: Flujo de sucesos, reporte horario profesor



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

- **Reporte Horarios**

Para la consulta del horario de un grupo se seguirá el siguiente procedimiento:

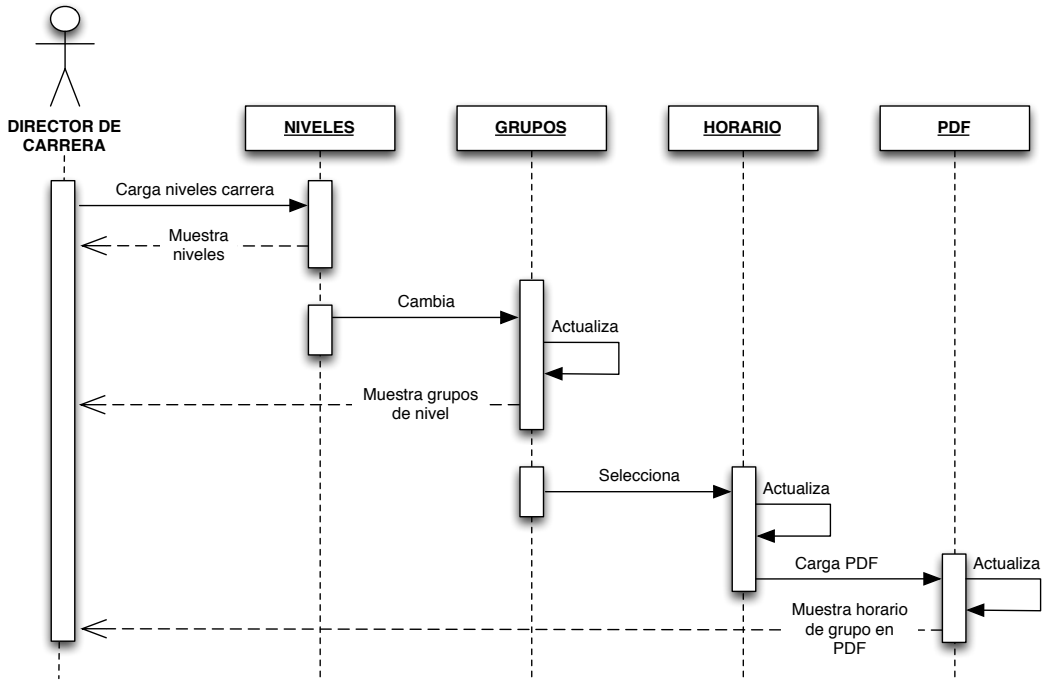
1. Debe seleccionar un nivel de una lista de ellos (serán únicamente los niveles que corresponden a la carrera que el director pertenece), ya que no podrá ver otro objeto más.

2. Debe escoger un grupo, de un listado de ellos, pertenecientes al nivel seleccionado en el paso 1, por lo que no van a aparecer grupos mientras no haya realizado el paso anterior.

   Con el segundo paso realizado se mostrará el horario en la Web en formato PDF.

Lo mencionado se puede observar el la figura 29.

Figura 29: Flujo de sucesos, generación de reportes



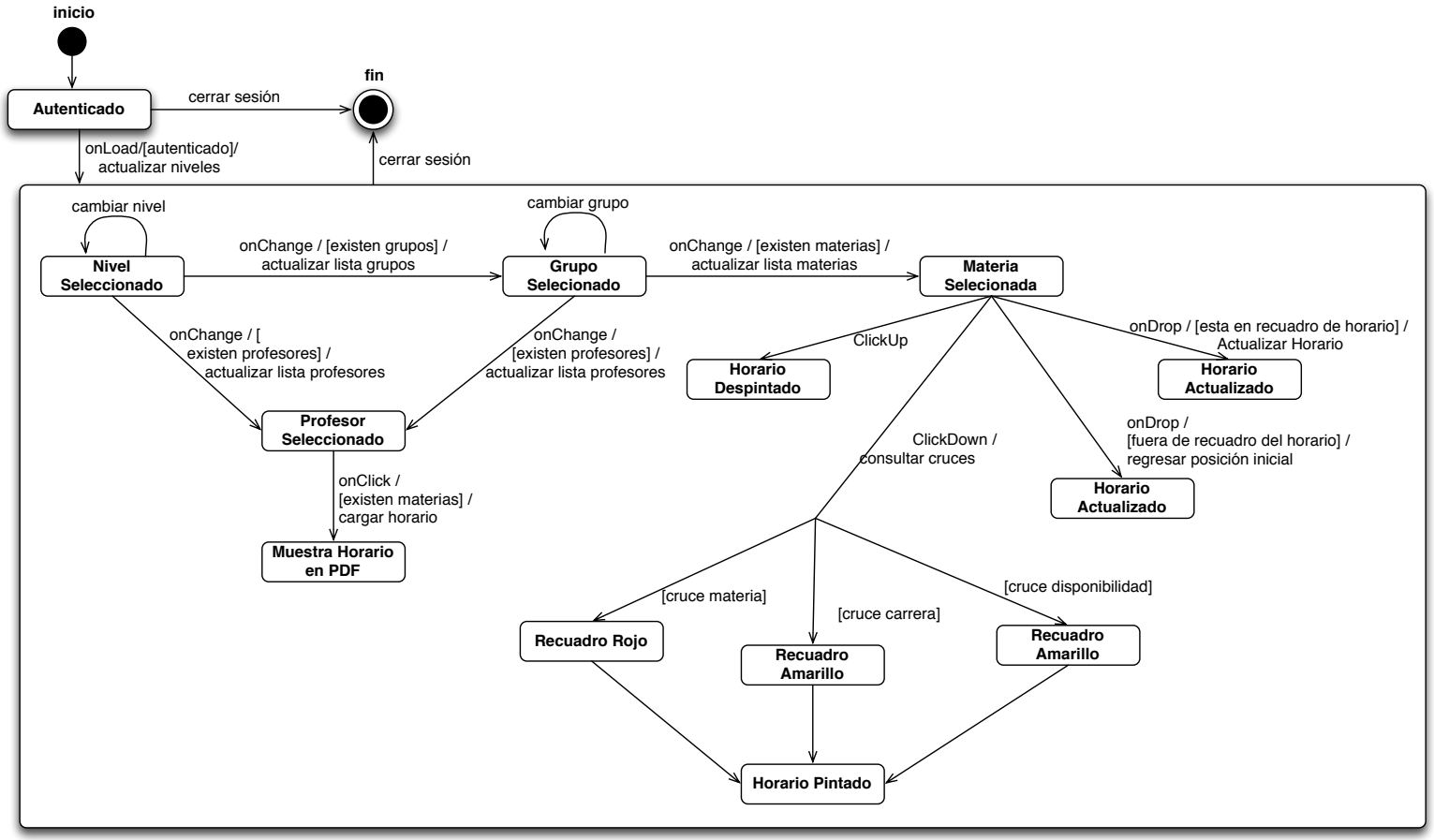
Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Diagrama de Estados

Los estados por los que pasara el sistema para la gestión de horarios será la siguiente: cuando el director de carrera se haya autenticado, los niveles cargaran; si un nivel cambia su estado ha seleccionado, este cargara niveles y profesores; en el caso de seleccionar un profesor se mostrará el horario, y si se escoge un grupo las materias de ese grupo cargaran; luego con la selección de una materia el horario se pintara con cruces (ya sea amarillo, rojo o gris) y se lo arrastra a otra posición entonces será el horario actualizado

Una vez autenticado el director de carrera, podrá cerrar sesión en cualquier momento. Esto se muestra en la figura 30.

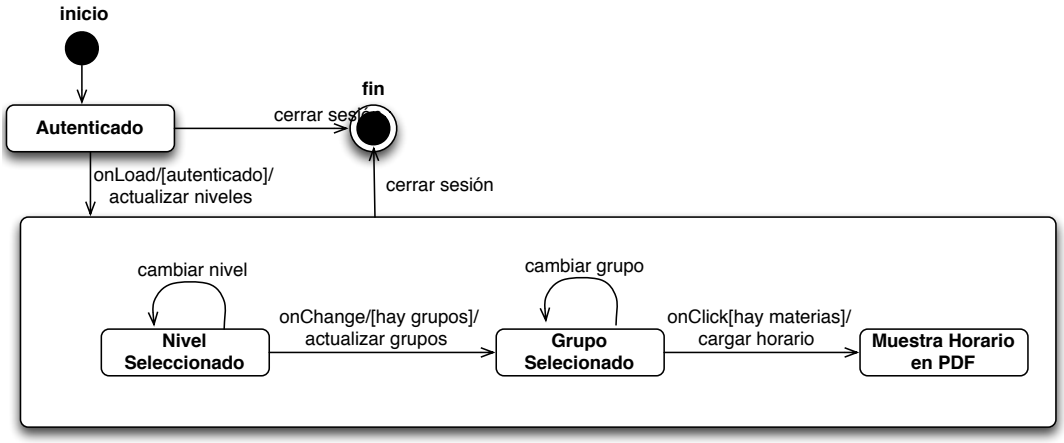
Figura 30: Diagrama de estados, gestión de horarios



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Para la creación de reportes el sistema pasará por los siguientes estados: cuando el director de carrera este autenticado, los niveles correspondientes a su carrera se cargarán, tomando en cuenta que podrá cerrar su sesión en cualquier momento, cuando un nivel es seleccionado los grupos de este se cargarán, pasado esto se puede escoger un grupo, para así mostrar el horario del grupo escogido en formato PDF. Lo mencionado anteriormente se puede observar en la figura 32.

Figura 31: Diagrama de estados, generación de reportes



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

2.4.1.4. Modelo funcional

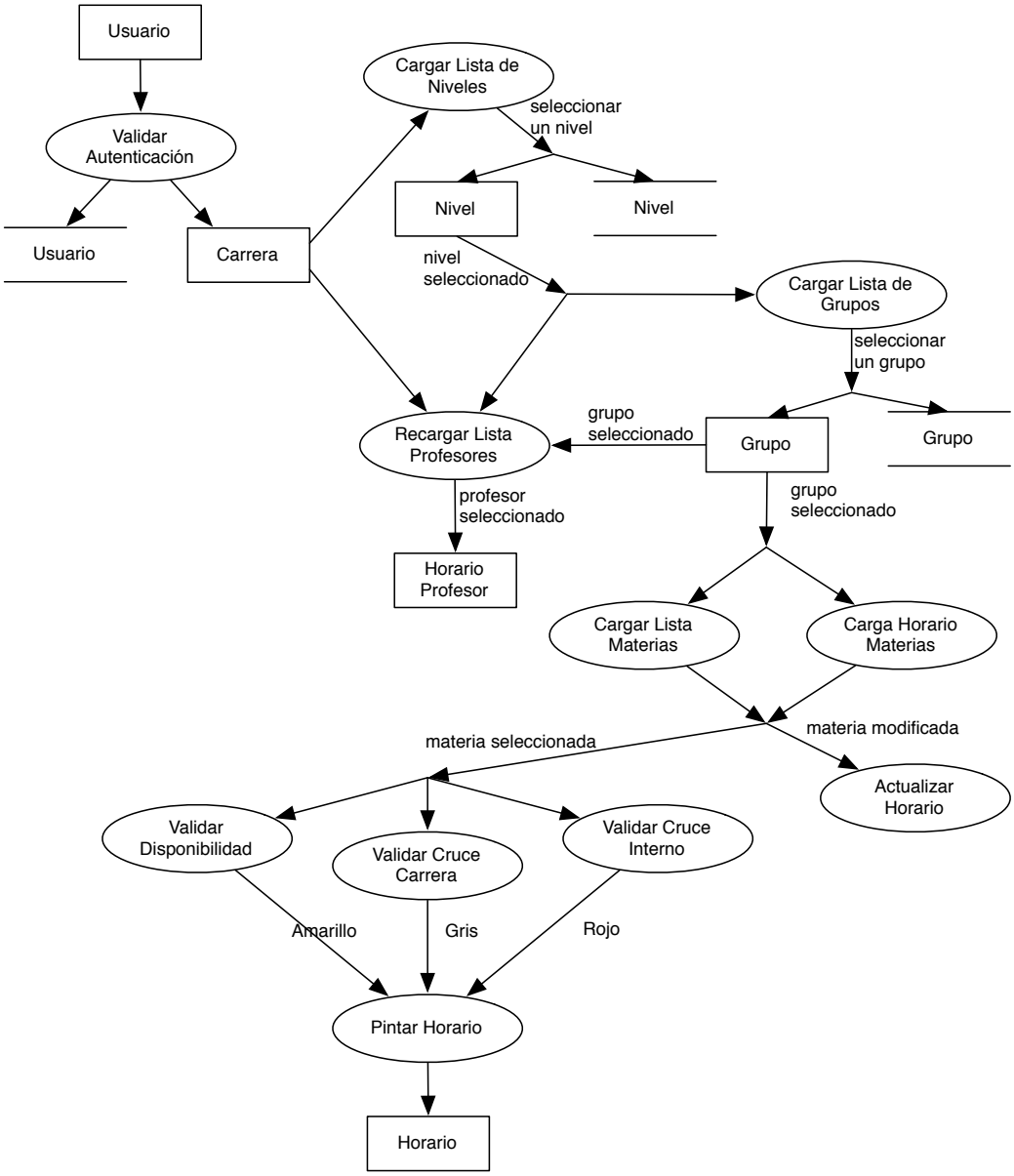
En la figura 32 se muestra el sistema para la generación de horarios, con sus objetos y procesos, para su correcto funcionamiento.

En este se ve al usuario, una vez validado, podrá ver los profesores, y podrá seleccionar un nivel, quedando este en memoria y perteneciente a la carrera a la cual pertenece el director; luego de esto seleccionara un grupo, el cual también quedara en memoria.

Cuando este seleccionado el grupo, se podrá escoger una materia, esto validará cruces mostrando en el horario; y en el caso de cambiar de posición a la materia el horario será actualizado.

Ya sea cuando se escoja un nivel o grupo, lista de profesores será recargada.

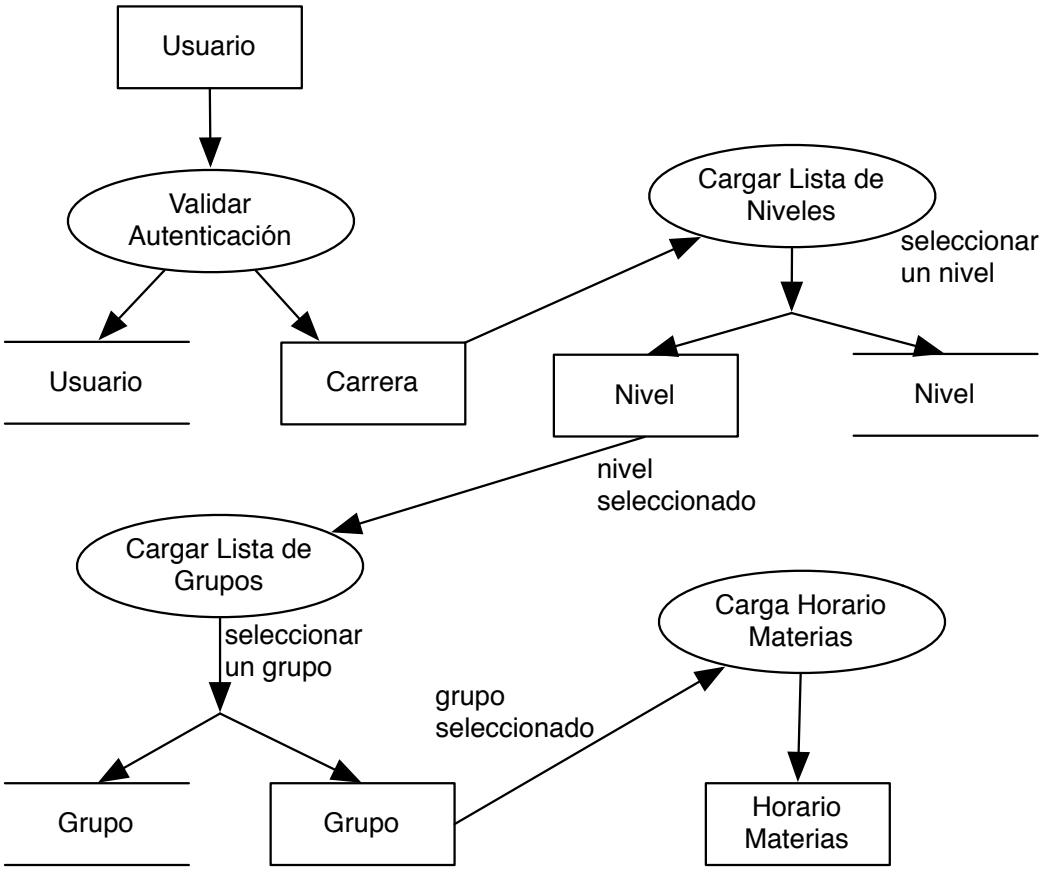
Figura 32: Modelo funcional, gestión de horarios



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

Para la generación de reportes la funcionalidad será la siguiente. Para poder ver el horario de un grupo, se valida autenticación, con esto se puede saber a la carrera que pertenece el usuario, y cargará los niveles pertenecientes a la carrera, al seleccionar un nivel, se puede escoger un grupo que cargara los horarios. Para esto quedará en memoria la carrera, nivel y grupo, para poder hacer la consulta y mostrar la información correcta. Esto se observar en la figura 33.

Figura 33: Modelo funcional, generación de reportes



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

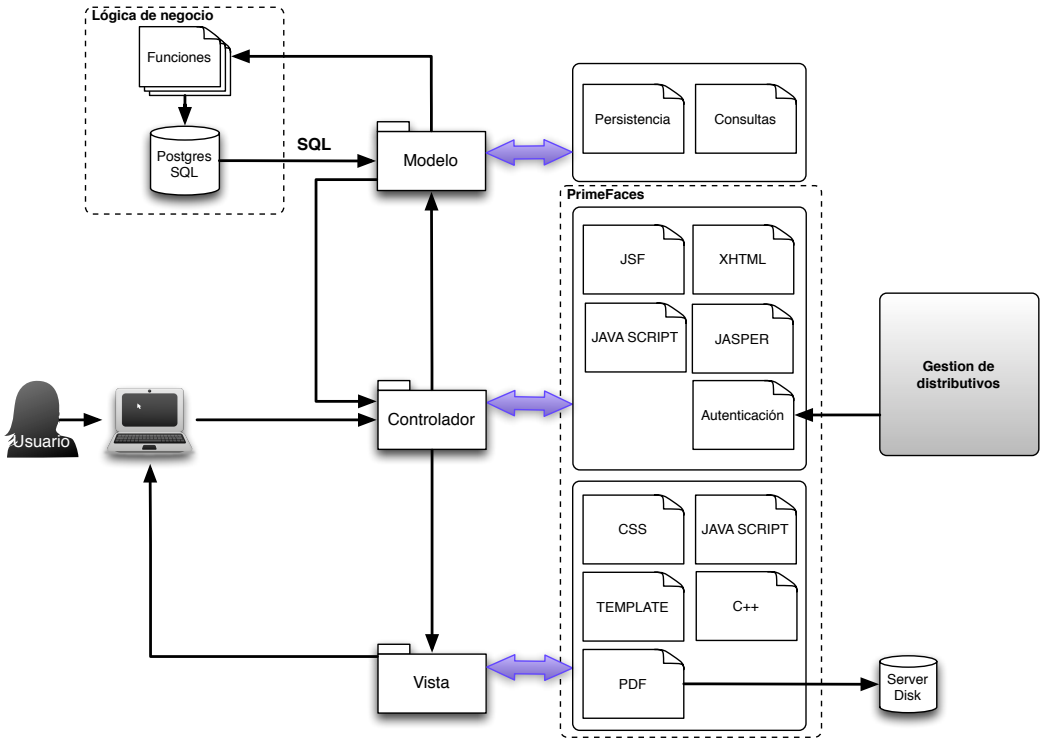
2.4.2. Diseño

2.4.2.1. Diseño del Sistema

En base en la arquitectura "MVC" (Aguilar & Martinez, 2011, pp. 1-100), se ha realizado el diseño del sistema. Separando la lógica del negocio en el modelo y enfocando a esta en la base de datos, mediante funciones.

Dentro del sistema se usa solamente lógica de aplicación; no se mezcla con la lógica de negocio, para el mantenimiento sea más eficiente. Si llegara a existir cambios en el estructura de la base de datos o nuevas reglas de negocio los cambios serian independientes y en línea, dejando de depender con el funcionamiento del sistema. Este diseño se ilustra en la figura 34.

Figura 34: Diseño del Sistema



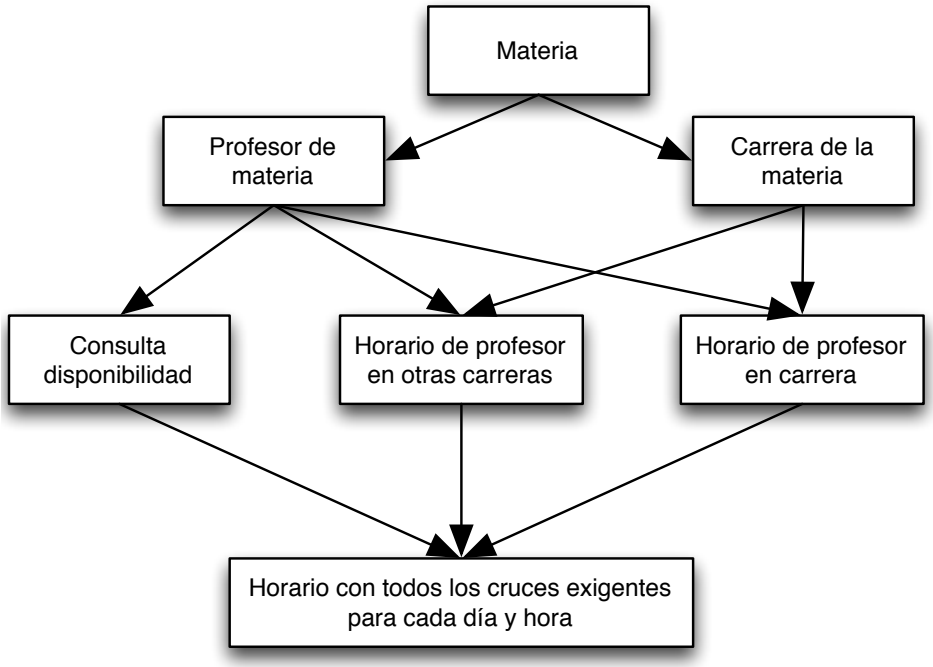
Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

2.4.2.2. Diseño de Objetos

En base a la figura 35 de modelo de objetos, se extrajo las funciones más importantes donde se basa de la lógica de negocio, para así poder entender y crear los algoritmos que darán funcionamiento al sistema; se ha creado los siguientes diagrama para poder exponer la solución encontrada.

En la figura 36 muestra la solución para validar los 3 tipos de cruce que se requirieron.

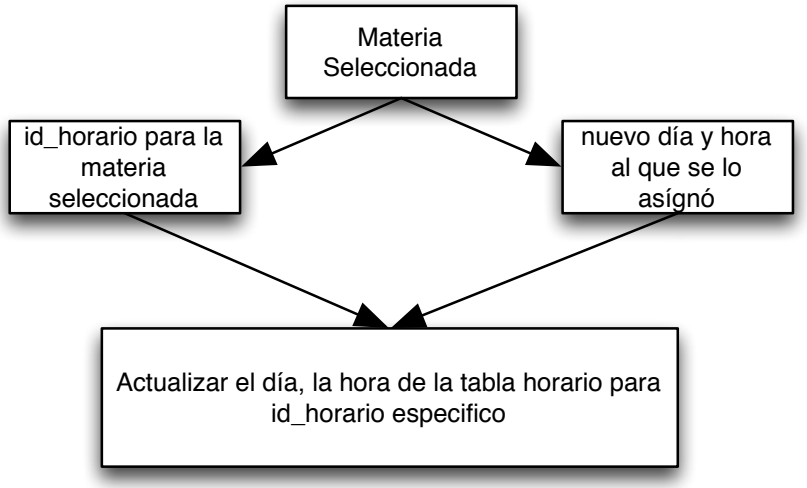
Figura 35: Diseño de Objetos para la validación de cruces



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

La solución para actualización de datos en la tabla principal (tbl_horarios) para el manejo de horarios se muestra en la figura 36.

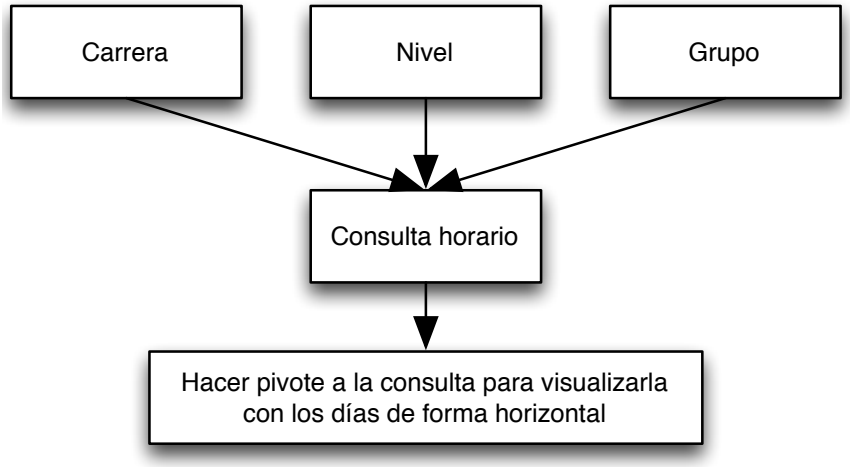
Figura 36: Diseño de Objetos para la validación de cruces



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

La figura 37, muestra la manera en que se expondrá el horario de una forma amigable, mediante una consulta a la base, teniendo los acoples con las tablas intervinientes, necesario para no duplicar o mostrar información errónea.

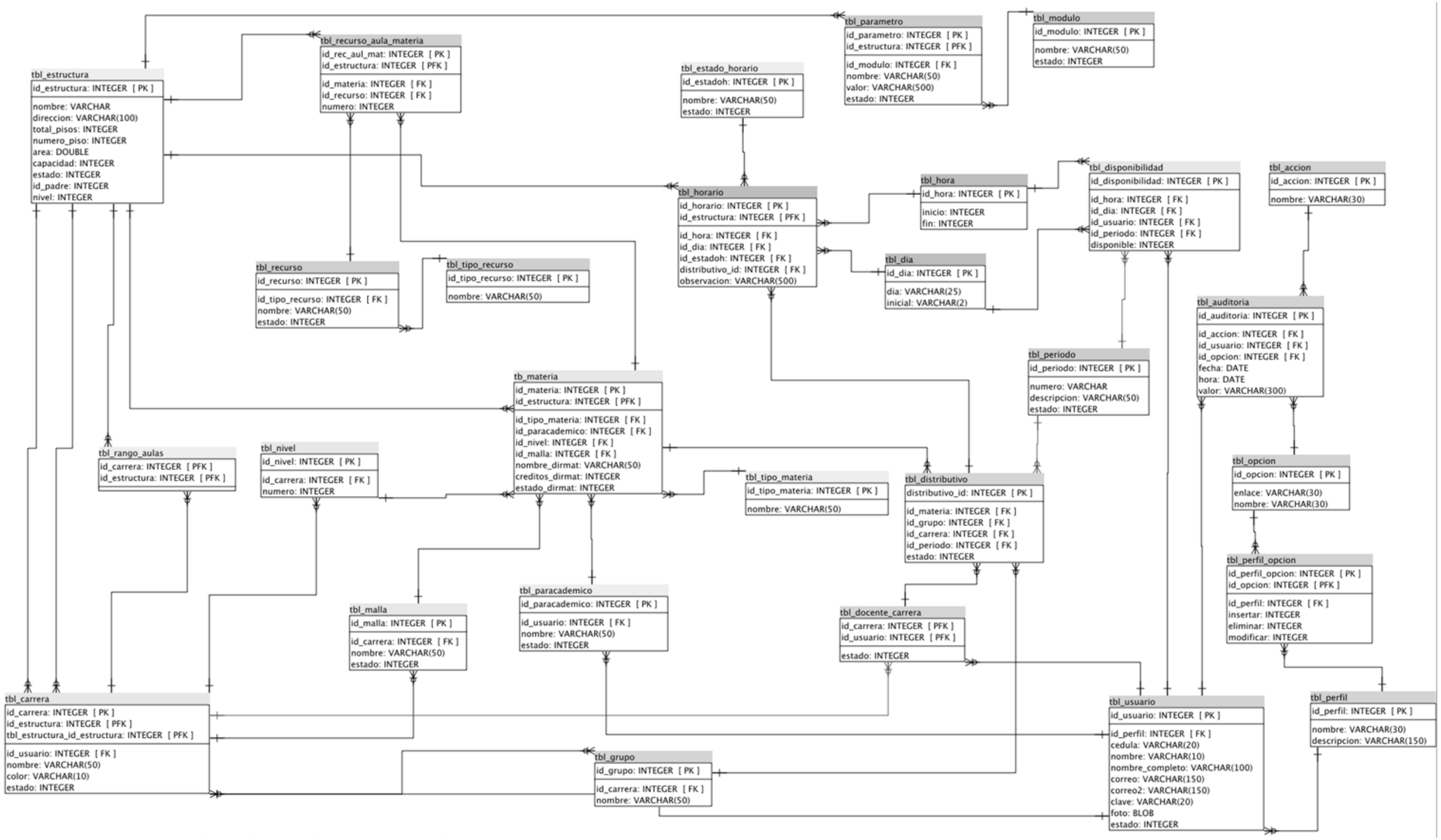
Figura 37: Diseño de Objetos para la validación de cruces



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

2.5. Diagrama Conceptual de la Base de Datos

Figura 38: Diseño de Objetos para la validación de cruces



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

CAPÍTULO 3
CONSTRUCCIÓN

3.1. Plataforma

El sistema de generación de horarios es multiplataforma, gracias a que cuando un programa Java es compilado en Unix, puede ser ejecutado en cualquier sistema operativo, ya sea Macintosh, Windows, Linux, entre otros; porque no depende del lenguaje de máquina del CPU, ya que maneja una máquina virtual Java (JVM), esta máquina virtual Java es quien interpreta y ejecuta el código Java, siendo traducido a Bytecode. Existe muchas JVM, una para cada tipo de CPU y Sistema Operativo.

Para entender mejor, una plataforma es una combinación de arquitectura de la CPU y sistema operativo. Tomando en cuenta esto, se ha decidió desarrollar el sistema con:

- Glassfish, como servidor de aplicaciones.
- JDK, como herramienta para desarrollo de Java.
- PostgresSQL, como servidor de base de datos.

Transformando estas herramientas en la plataforma real del módulo de gestión de horarios, las cuales trabajan bajo arquitecturas de CPU : x86 , x86_64 , IA64 , PowerPC , PowerPC 64, S/390 , S/390x , Sparc , Sparc 64 , Alpha , ARM, MIPS , Mipsel , M68K y PA -RISC. Y funcionar en varias plataformas de sistemas operativos como: Windows, Linux, Unix, Mac, AIXy y Mac OS X; haciéndole más potente el módulo de generación de horarios.

3.2. Lenguaje de Programación

3.2.1. Java

Las características que tiene Java son: sencillez, orientado a objetos, robusto, seguro, arquitectura neutra, alto rendimiento, dinámico, libre de licencia y sobre todo portable. Características que lo diferencian de otros lenguajes.

3.2.2. JavaScript

JavaScript es un lenguaje muy sencillo de usar; su sintaxis es muy similar a Java. JavaScript y Java son diferentes vistos desde la forma en la que interactúa con la CPU ya que Java necesita ser compilado, mientras que JavaScript es interpretado por los navegadores línea por línea, cumpliendo el objetivo de auxiliar a HTML.

3.2.3. XHTML

Es una evolución del HTML con características de XML, haciéndolo más robusto. Durante el desarrollo no se puede dejar tags sin cerrar como ocurría con HTML, ya que su estructura es rígida y no deja interpretaciones al navegador, para evitar el consumo recursos del lado cliente. XHTML da prioridad a los objetos sin enfatizar la parte visual.

3.2.4. XML

Lenguaje de marcas generalizado; es un lenguaje que usa sus propias marcaciones o tags, estructurando la información en documentos que contengan texto.

El lenguaje XML se lo puede dar uso en: la Web, para intercambio de información entre diferentes instancias de forma segura, fiable y libre, estructurar bases de datos, entre otros.

Se debe tomar en cuenta que es sensible a la escritura, es decir, diferencia los caracteres mayúsculas y minúsculas.

3.2.5. SQL

Lenguaje de Consulta Estructurado es un lenguaje estandarizado por las normas ANSI/ISO, vinculadas principalmente a la definición, manipulación y gestión de base de datos relacionales, convirtiéndose en uno de los lenguajes de base de datos más usado. Las características que lo volvieron tan popular son que permite la utilización de algebra y calculo relacional.

La funcionalidad de este lenguaje basado en sentencias es la de obtener información de una base de datos de forma ágil, fácil y rápida. Lo que lo vuelve tan sencillo es su similitud a un lenguaje natural (Ingles).

3.2.6. CSS

El lenguaje de hojas de estilo es usado para definir el aspecto y formato de un documento con lenguaje de marcas. Puede ser usado para definir la presentación de archivos de formato HTML, XHTML, XML, entre otros.

3.3. Herramientas y Librerías utilizadas

3.3.1. JDK

Esta herramienta fue realizada por la empresa Sun Microsystems para crear compilar y ejecutar aplicaciones Java, se la puede conseguir de forma gratuita en su portal Web en su página oficial, cada JDK viene con su versión Java.

Es importante la instalación de esta herramienta; ya que dependen de el para el funcionamiento de Netbeans, Eclipse, PostgresSQL, JasperReport y Glassfish.

3.3.2. Netbeans

NetBeans es un entorno de desarrollo integrado (IDE) para Java; pertenece a la comunidad NetBeans. Este entorno permite escribir, compilar, depurar y ejecutar programas. En la actualidad existen dos productos: NetBeans IDE, producto sin restricciones de uso y Netbeans Platform, base modular o extensible.

Los dos productos de NetBeans son de código abierto y gratuito; su código fuente está disponible para reutilización en su sitio Web.

3.3.3. Eclipse

Es un entorno de desarrollo integrado (IDE) para Java, de código abierto; esta herramienta gratuita se encuentra disponible en su página oficial. Este IDE es desarrollado en Java.

Fue desarrollado por IBM y actualmente pertenece a la fundación Eclipse (comunidad de código abierto).

Al igual que Netbeans, existen dos productos: Eclipse IDE for Java EE Developers (para entornos profesionales y empresariales) y Eclipse IDE for Java Developer (para entornos de programación SE de Java).

3.3.4. JasperReport

JasperReport es una librera desarrollada en Java, con licencia GNU y perteneciente a JasperSoft. El uso de esta es para la generación de informes, maneja en formato PDF, CSV, XML, TXT, HTML, XML, RTF y Jasper viewer.

El fichero base para la generación de informes se lo hace en XML, a este se lo compila para generar un archivo .jasper, con el que interactúa la aplicación.

Para que el desarrollo sea más amigable se tiene interfaces como: IDE iReport o agregar plugin JasperReport a NetBeans; teniendo las misma funcionalidad que con iReport.

3.3.5. PostgreSQL

Es un aplicativo para la administración de Base de Datos. Esta herramienta es de licencia libre; su IDE para administración es PgAdmin.

También es una herramienta que trabaja con relación y herencia entre tablas, a lo que se lo conoce como objetos relacionales.

A pesar de manejar su propio lenguaje PL/PgSQL su sintaxis SQL es estándar por lo que es fácil de aprender, ya que existe mucha información en la Web; en este se puede usar datos de tipo: fecha, monetarios, elementos gráficos, datos sobre redes, cadenas de bits, entre otros; pero algo que llama la atención es la creación de tipos de datos propios.

3.3.6. Glassfish

Es un servidor de aplicaciones de código abierto, desarrollada por la empresa Sun Microsystems al igual que Java; ahora es perteneciente a Oracle.

Glassfish es un derivado de Apache Tomcat (Apache también es un servidor de aplicaciones), usando Grizzly que es para escalabilidad y velocidad.

Su IDE de administración se lo puede manejar desde JDeveloper, IntelliJ, Eclipse y Netbeans; permitiendo administrarlo desde un navegador Web.

3.3.7. PrimeFaces

PrimeFaces es una librería de código abierto para JSF 2.0 desarrollada y mantenida por Prime Technology. PrimeFaces tiene un conjunto de componentes visuales, esta librería es compatible con RichFaces (otra librería similar), es decir se pueden trabajar en conjunto; una gran ventaja de PrimeFaces es que lleva integrado Ajax.

PrimeFaces ayuda en la creación de las aplicaciones Web, ya sea para computadores como para dispositivos móviles.

El entendimiento es sencillo ya que tiene muy buena documentación y ejemplos con código, tanto en su documento de guía de usuario como en su página Web oficial.

3.4. Implementación de Base de Datos

A continuación se describe el diccionario de datos, con las tablas que interviene el "módulo de Gestión de Horarios"

Tabla 5: Tabla diccionario de datos base de datos

tbl_dia	Tabla	Información de días
id_dia	Integer	Identificador de la tabla día
dia	Character varying(25)	Nombre del día
inicial	Character varying(10)	Abreviatura del día
tbl_hora	Tabla	Información de las horas del horario
id_hora	Integer	Identificativo de la tabla hora
inicio	Integer	Descripción del inicio de la hora
fin	Integer	Descripción del fin de la hora
tbl_nivel	Tabla	Información de niveles
id_nivel	Integer	Identificativo de la tabla nivel
numero	Integer	Numero de nivel de la tabla nivel
nombre	Character varying(50)	Descripción del número de nivel de la tabla nivel
tbl_grupo	Tabla	Información de los grupos
id_grupo	Integer	Identificativo de la tabla grupo
id_carrera	Integer	Identificativo de la tabla carrera, relacionada con la tabla carrera
nombre	Character varying(50)	Descripción del grupo
tbl_docente_carrera	Tabla	Información para relacionar la tabla docente y la tabla carrera
id_carrera	Integer	Identificativo de la tabla carrera, relacionada con la tabla carrera
id_usuario	Integer	Clave primaria de la tabla tb_sis_usuario
estado	Integer	Campo que indica si está disponible o no el docente en la carrera
tbl_período	Tabla	Información de períodos lectivos
id_período	Integer	Identificativo de la tabla período
numero	Character varying	Numero de período
descripcion	Character varying(50)	Descripción del período
estado	Integer	Estado de período, activo o inactivo
tbl_estado_horario	Tabla	Información del estado de un horario, para identificar el tipo de cruce
id_estadoh	Integer	Identificativo de la tabla estado horario
nombre character	Character varying(50)	Descripción de la tabla estado horario
estado	Integer	Campo que indica si está habilitado o deshabilitado el registro
color	Character varying(10)	Color que identifica el estado de cruce

tbl_carrera	Tabla	Información de carreras
id_carrera	Integer	Identificador de la tabla carrera
id_campus	Integer	El campus al cual pertenece la carrera
id_directo	Integer	Clave foránea de la tabla tb_usuario. Representa al director de carrera
nombre	Character varying(100)	Nombre de la carrera
color	Character varying(10)	Color identificativo de la carrera
estado	Integer	Carrera en estado activo o inactivo
tbl_disponibilidad	Tabla	Información de disponibilidad de docente
id_disponibilidad	Integer	Identificativo de la tabla disponibilidad
id_hora	Integer	Identificativo de hora relacionada con la tabla hora
id_dia	Integer	Identificativo de la tabla día, relacionada con la tabla día
id_docente	Integer	Identificativo del docente, relacionada con la tabla usuario
id_período	Integer	Identificativo del período, relacionada con la tabla período
disponible	Integer	Campo que indica si está disponible o no el docente
tbl_distributivo	Tabla	Información del distributivo
id_distributivo	Integer	Identificativo de la tabla distributivo
id_carrera	Integer	Identificativo de la tabla carrera, relacionada con la tabla carrera
id_docente	Integer	Identificativo del docente, relacionada con la tabla usuario
id_materia	Integer	Identificativo de la tabla materia, relacionada con la tabla materia
id_grupo	Integer	Identificativo de la tabla grupo, relacionado con la tabla grupo
id_período	Integer	Identificativo del período, relacionada con la tabla período
estado	Integer	Campo que indica si está disponible o no el distributivo
tbl_horario	Tabla	Información del horario, tabla principal del módulo
id_horario	Integer	Identificativo de la tabla horario
id_carrera	Integer	Identificativo de la tabla carrera, relacionada con la tabla carrera
id_estadoh	Integer	Identificativo de la tabla estado horario relacionado con la tabla estado horario
id_distributivo	Integer	Identificativo de la tabla distributivo relacionado con la tabla distributivo
id_aula	Integer	Identificativo de la tabla aula relacionada con la tabla aula
id_hora	Integer	Identificativo de la tabla hora, relacionado con la tabla hora

id_dia	Integer	Identificador de la tabla día, relacionado con la tabla día
observacion	Character varying(500)	Observaciones del registro de la tabla horario
tbl_materia	Tabla	Información de la tabla materia
id_materia	Integer	Identificativo de la tabla materia
id_carrera	Integer	Identificativo de la tabla carrera, relacionada con la tabla carrera
id_paracademico	Integer	Si este campo es NULL, entonces la materia es regular o normal
id_nivel	Integer	Identificativo de la tabla nivel, relacionado con la tabla nivel
id_malla	Integer	Identificativo de la tabla malla, relacionado con la tabla malla
id_tipo_aula	Integer	Identificativo de la tabla tipo aula, relacionado con la tabla tipo aula
codigo	Integer	Código interno de las materias de la UPS
nombre	Character varying(50)	Descripción de la materia
creditos	Integer	Número de créditos de la materia
estado	Integer	Estado de materia, para activarla o desactivarla
tbl_usuario	Tabla	Información de usuarios del sistema
id_usuario	Integer	Identificativo de la tabla tb_sis_usuario
id_perfil	Integer	Identificativo de la tabla perfil relacionado con la tabla perfil
estado_sesion	Integer	Campo para validar el estado de la sesión de usuario
cedula	Character varying(20)	Cedula o pasaporte del usuario
nombre	Character varying(10)	Nombre con el que se realiza el login
nombre_completo	Character varying(100)	Nombre real del usuario, servirá para generación de reportes y como información complementaria
correo	Character varying(150)	Correo institucional del usuario
correo2	Character varying(150)	Correo personal del usuario
clave	Character varying(200)	Clave con la que se realiza el login
tema	Character varying(50)	Interfaz gráfica que el usuario desea utilizar
foto	Character varying(50)	Almacena la foto del usuario

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

3.5. Codificación

Mientras se va generando código, es necesario comentar: qué, y para qué, se va a implementar una línea de código; ya sea una clase, un método e incluso una variable (comentar las más importantes, y las que se considere necesiten de una explicación). El motivo por el que se debe comentar, es hacer más comprensible el código al desarrollador, teniendo la visión de que un futuro manipularan nuevos desarrolladores; para que así el crecimiento de la aplicación no dependa solo del programador inicial.

Se debe tomar en cuenta que la implementación del sistema maneja 2 lógicas: la lógica de la aplicación (desarrollada en JSF) y la lógica del negocio (a nivel de base de datos mediante funciones en PostgreSQL).

3.5.1. Lógica de Aplicación

A continuación se muestra y explica las clases, funciones, métodos y variables más importantes para el correcto funcionamiento del sistema.

La clase “gestion_horarios.java” es la más importante del módulo, ya que interactúa con el navegador Web; para el envío y recepción de información tiene variables globales mostradas a continuación:

```
// <editor-fold defaultstate="collapsed" desc=" Region Variables Globales ">
public List<tabla_horario> horario_profesor;
private List<Materia> ListaMaterias;
private List<Materia_sin_Horario> ListaMaterias_sin_horario;
private List<Materia> id_materia_escogida;
private String tab_grupo_anterior;
private String tab_grupo_actual;
private List<Dia_Hora_Cruce> LisraDia_Hora_Cruce;
// ================================== FILTROS DE CONSULTA
// ======================================
private String carrera;
private String nivel;
private String grupo;
private String path;
private Map<String, Integer> m_carreras = new HashMap<String, Integer>();
private Map<String, Integer> m_niveles = new HashMap<String, Integer>();
private Map<String, Integer> m_grupos = new HashMap<String, Integer>();
private List<Grupo> tabs;

// </editor-fold>
```

Cada una de estas variables necesita sus métodos getter y setter para que el archivo xhtml pueda leerlos; esto se puede ver a continuación:

```
public void setNivel(String nivel) {
	this.nivel = nivel;
}
public void setGrupo(String grupo) {
	this.grupo = grupo;

}


public String getNivel() {


	return nivel;
}
public String getGrupo() {
	return grupo;

}
```

Los métodos esenciales, con los que se interactúa con el navegador y la Clase son:

onDrop()

```
public void onDrop(DragDropEvent event) {
	FacesContext context = FacesContext.getCurrentInstance();
	if (actualizar_horario(obteneID_horario(event.getDragId()),
	obteneID_dia(event.getDropId()),
	obteneID_hora(event.getDropId())))
	context.addMessage(null, new FacesMessage("Satisfactorio","HORARIO
MODIFICADO"));
	else
		context.addMessage(null, new FacesMessage("Error","HORARIO NO
MODIFICADO"));

}
```

Se ejecuta una vez que se ha modificado el horario, llamando a la función "actualizar_horario()" que se encarga actualizar el horario. La parte más importante es saber que id_horario se ha cambiado de día y hora; ya que en base a estos llama a la función de PostgreSQL encargada de hacer el update respectivo.

Down()

```
public void down(ActionEvent event) {
        try {
        FacesContext context =
FacesContext.getCurrentInstance();
        Map<String, String> map =
context.getExternalContext().getRequestParameterMap();
        String value = (String) map.get("id_profesor_");

        RequestContext requestContext =
RequestContext.getCurrentInstance();
        LisraDia_Hora_Cruce = new
ArrayList<Dia_Hora_Cruce>();
        LisraDia_Hora_Cruce =
consulta_cruce.consulta_cruce_horario(
                                obteneID_profesor(value),
Integer.parseInt(carrera));
        for (int i = 0; i < LisraDia_Hora_Cruce.size(); i++)
{
                requestContext.execute("CambiaColorFondo('#"+
                        retornaDia(LisraDia_Hora_Cruce.get(i).
                        getId_dia()).toString() +
                        "-" +
                        LisraDia_Hora_Cruce.get(i).getId_hora(
                        ) +
                        "','" +
                        LisraDia_Hora_Cruce.get(i).getColor()
                        + "')");
        }
        } catch (Exception e) {
                // enviar a paguina de error
        }

}
```

Obtiene el id del profesor que dicta la materia seleccionada.

Consulta horario actual del profesor con sus tipos de cruce.

Ejecuta función javascript en el navegador, para que coloree el horario en base a la consulta efectuada.

El archivo “gestión_horarios.xhtml”, es la página Web que interactúa con el usuario y este depende de la clase “gestión_horarios.java” anteriormente mencionada. Aquí se encuentran funciones javascript; las más importantes son:

ResetarFondoColoresHorario()

```
function  ResetearFondoColoresHorario(){
        $("#LUNES-1").css("background","#FFFFFF");
        $("#MARTES-1").css("background","#FFFFFF");
        $("#MIERCOLES-1").css("background","#FFFFFF");
        $("#JUEVES-1").css("background","#FFFFFF");
        $("#VIERNES-1").css("background","#FFFFFF");
        $("#SABADO-1").css("background","#FFFFFF");
        $("#DOMINGO-1").css("background","#FFFFFF");

        $("#LUNES-2").css("background","#FFFFFF");
        $("#MARTES-2").css("background","#FFFFFF");
        $("#MIERCOLES-2").css("background","#FFFFFF");
        $("#JUEVES-2").css("background","#FFFFFF");
        $("#VIERNES-2").css("background","#FFFFFF");
        $("#SABADO-2").css("background","#FFFFFF");
        $("#DOMINGO-2").css("background","#FFFFFF");
```

```
        $("#LUNES-3").css("background","#FFFFFF");
        $("#MARTES-3").css("background","#FFFFFF");
        $("#MIERCOLES-3").css("background","#FFFFFF");
        $("#JUEVES-3").css("background","#FFFFFF");
        $("#VIERNES-3").css("background","#FFFFFF");
        $("#SABADO-3").css("background","#FFFFFF");
        $("#DOMINGO-3").css("background","#FFFFFF");
}
```

Esta función se ejecuta cuando se suelta el clic del objeto materia., mediante esta función se pinta todos los recuadros del horario de fondo blanco (color inicial que explica que no hay cruces).

CambiarColorFondo()

```
function CambiaColorFondo(id_objeto,color)
{
        $(id_objeto).css("background",color);
}
```

Esta función se encarga de pintar a cada recuadro del horario del color que le corresponda según el cruce; y es ejecutado desde el método "down()" de la clase "gestión_horarios.java".

Materia_mouseDown()

```
function materia_mouseDown( id_)
{
        remoteCommandFunctionDown([{name:'id_profesor_',value:id_}]);

}
```

A pesar de ser una función sencilla, es una de las más importantes, ya que esta se encarga de pasar el id del profesor que dicta la materia seleccionada. En base a esta consulta los cruces y los manda a pintar con los métodos descritos anteriormente.

Dentro del archivo "gestión_horarios.xhtml", hay una parte que puede ser confusa; esta es para la carga dinámica de las materias en cada recuadro del horario.

Para entenderlo, primero se carga la lista del horario en el objeto m_lun1 (para los otros recuadro varía el nombre).

```
<p:outputPanel id="LUNES-1" styleClass="slot">
```

"LUNES-1", será el recuadro del horario.

```
  <c:forEach var="m_lun1"
  items="#{gestion_horarios.listaMaterias}">
```

"m_lun1", tendrá la lista de materias, con sus datos; y se irá recorriendo.

```
    <c:set var="lun1"
    value="#{m_lun1.id_dia}_#{m_lun1.id_Hora}" />
```

"lun1", ira variando el día y hora, de la materia.

```
     <c:if test="${lun1 == '1_1'}">
      <p:panel id="_#{m_lun1.id_Horario}"
      styleClass="recuadro_materia">
        <p:draggable revert="true" />
        <div id="m_#{m_lun1.id_profesor}"
        mouseup="materia_mouseup(this.id);"
        onmousedown="materia_mouseDown(this.id);"
        class="div_recuadro_materia">
            <h:panelGrid columns="2"
            styleClass="panelGrid_div">
              <h:outputText
             styleClass="texto_materia_codigo"
              value="#{m_lun1.getcod_materia()}" />
              <h:outputText styleClass="texto_materia"
              value="#{m_lun1.nombreMateria}" />
            </h:panelGrid>
        </div>
      </p:panel>
    </c:if>
  </c:forEach>
```

Cuando "lun1" sea igual a 1_1 cargará el panel de la materia con sus datos; para martes tercera hora ("mar3") igualara con 2_3, y así para el resto del horario.

```
  <p:droppable tolerance="fit"
activeStyleClass="slotActive"
               onDrop="handleDrop">
       <p:ajax listener="#{gestion_horarios.onDrop}"
                update="growl" />
  </p:droppable>
</p:outputPanel>
```

Cada objeto manejando PrimeFaces para el control del drag drop.

3.5.2. Lógica de Negocio

Para el manejo de la lógica de negocio, se lo hace desde PostgreSQL, para optimizar tiempos aprovechando el motor de; las funciones más importantes son:

obtener_materias()

```
CREATE OR REPLACE FUNCTION obtener_materias(IN i_id_carrera integer, IN i_id_nivel
integer, IN i_id_grupo integer)

RETURNS TABLE(o_id_horario integer, o_id_distributivo integer, o_codigo_materia
integer, o_nombre_materia character varying, o_id_carrera integer, o_id_nivel integer,
o_id_grupo integer, o_id_profesor integer, o_nombre_profesor character varying,
o_período integer, o_dia integer, o_hora integer, o_cruce integer) AS $BODY$

BEGIN RETURN QUERY

select h.id_horario, d.id_distributivo, m.codigo, m.nombre, dc.id_carrera, m.id_nivel,
d.id_grupo, d.id_docente, u.nombre_completo,
d.id_período,h.id_dia,h.id_hora,h.id_estadoh              from tbl_distributivo d
                        inner join tbl_docente_carrera dc on dc.id_usuario =
                d.id_docente
                        and dc.id_carrera = d.id_carrera
                        inner join tbl_materia m  on m.id_materia = d.id_materia
                        and m.id_carrera = d.id_carrera and dc.id_carrera =
                m.id_carrera          inner join tbl_usuario u on u.id_usuario =
                d.id_docente
                        and dc.id_usuario = u.id_usuario
                        inner join tbl_grupo g on  d.id_grupo = g.id_grupo
                        and g.id_carrera = d.id_carrera
                        inner join tbl_horario h on h.id_distributivo =
                d.id_distributivo
                where d.id_carrera = i_id_carrera and m.id_nivel = i_id_nivel
                        and d.id_grupo = i_id_grupo and h.id_hora <> 0
                        and h.id_dia <> 0
order by h.id_dia, h.id_hora;
END;
$BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;
ALTER FUNCTION obtener_materias(integer, integer, integer)
OWNER TO postgres;
```

Esta función devuelve una tabla con la información de horario de un nivel, carrera y grupo; pero solo de las materias que ya han sido posicionadas en un recuadro del horario, para las materias que aún no lo han sido se tiene otra función similar; esta es obtener_materias_sin_horario().

Obtener_cruce_profesor()

Esta consulta devolverá el horario del profesor con sus cruces, es necesario 2 parámetros; el id_profesor, necesario para saber a quién pertenece el horario; y id_carrera para saber si es de su carrera o no (controlar tipo cruce carrera).

```
CREATE OR REPLACE FUNCTION obtener_cruce_profesor(IN i_id_profesor integer, IN
i_id_carrera integer)
RETURNS TABLE(o_id_dia integer, o_id_hora integer, o_id_cruce integer, o_cruce
character varying, o_color character varying)
AS $BODY$ BEGIN
```

Empieza guardando en una tabla temporal la disponibilidad del profesor.

```
CREATE TEMP TABLE temp_cruces_total ON COMMIT DROP AS
       select id_dia, id_hora, disponible
              from tbl_disponibilidad d
              where id_docente= i_id_profesor;
```

Luego se encuentra el horario del profesor, por lo que se obtiene los días y horas ya ocupados, agrupándolo por carrera; por lo que genera una tabla con tipo de cruce interno, como el de carrera.

```
CREATE TEMP TABLE temp_cruces ON COMMIT DROP AS
       select u.id_usuario, u.nombre, h.id_dia as dia, h.id_hora as hora,
               d.id_carrera, cast(count(u.id_usuario) as integer) as
               cantidad

              from tbl_distributivo d
              inner join tbl_usuario u on u.id_usuario = d.id_docente
              inner join tbl_horario h on h.id_distributivo =
              d.id_distributivo

              where h.id_hora <> 0 and h.id_dia <> 0
                     and u.id_usuario = i_id_profesor

              group by u.id_usuario, u.nombre, d.id_período, h.id_dia, h.id_hora,
              d.id_carrera

       order by h.id_dia, h.id_hora;
```

Posteriormente se une las 2 consultas anteriores, con lo que ya se conoce, el horario del profesor de la carrera en la que está siendo modificado, como la del resto de carreras (en caso que hubiera) junto con su disponibilidad

```
update temp_cruces_total
       set disponible =  case when (disponible = 0 and cantidad >= 1
                                       and id_carrera = i_id_carrera)
                                  then 4
                                  when (cantidad >= 1 and id_carrera <>
                                         i_id_carrera)
                                  then 3
                                  when (disponible = 1 and cantidad >= 1 and
                                  id_carrera = i_id_carrera) then 2
                                  else 2
                           end
       from temp_cruces where id_dia = dia and id_hora = hora ;
```

La tabla temporal final que existirá es solo de los días y horas que ya está asignado, y que por lo tanto no debería ser reasignado. Pero como se debe pintar todo el horario, se une con una consulta que devuelve todos los días y horas que existe en el horario, haciendo match con la tabla temporal “temp_cruces_total” y actualizándolo en el caso de haberlo.

```
RETURN QUERY  select id_dia, id_hora, id_estadoh, nombre as estado, color from
temp_cruces_total  inner join tbl_estado_horario on disponible = id_estadoh --where
disponible >1 order by 1,2;   END; $BODY$   LANGUAGE plpgsql VOLATILE   COST 100
ROWS 1000; ALTER FUNCTION obtener_cruce_profesor(integer, integer)   OWNER TO
postgres;
```

Actualiza_horario

```
CREATE OR REPLACE FUNCTION actualiza_horario(i_id_horario integer, i_id_dia integer,
i_id_hora integer)
RETURNS character varying AS $BODY$

DECLARE   vv_estado varchar;

BEGIN  UPDATE tbl_horario
               SET id_hora = i_id_hora
               ,id_dia = i_id_dia
       WHERE id_horario = i_id_horario;

       RETURN vv_estado;
END; $BODY$
LANGUAGE plpgsql VOLATILE
COST 100;
ALTER FUNCTION actualiza_horario(integer, integer, integer)
OWNER TO postgres;
```

Esta función se encarga de actualizar el la tabla horario, que es sobre la que gira el módulo.

Obtener_horario_profesor()

```
CREATE OR REPLACE FUNCTION obtener_horario_profesor(IN i_id_profesor integer)

RETURNS TABLE(o_hora integer, o_lunes character varying, o_martes character varying,
o_miercoles character varying, o_jueves character varying, o_viernes character
varying, o_sabado character varying, o_domingo character varying) AS $BODY$
```

Se crea la función encargada de devolver el horario de un profesor.

```
BEGIN
CREATE TEMP TABLE temp_horario ON COMMIT DROP AS
select id_dia, id_hora, '':: character varying as nombre from tbl_hora, tbl_dia
order by 1,2;
delete from temp_horario where (id_dia = 0 or id_hora = 0);
```

Posterior a esto se creó una tabla temporal que contenga toda la información del horario; es decir todos los registros existentes en el horario, con cruce de día y hora.

```
CREATE TEMP TABLE temp_horario_prof ON COMMIT DROP AS
select distinct h.id_dia,h.id_hora, (m.nombre || '-' || c.nombre || '- nivel'
||m.id_nivel
       || '- grpupo ' || g.nombre) :: character varying  as nombre
from tbl_distributivo d
       inner join tbl_materia m        on m.id_materia = d.id_materia
       and m.id_carrera = d.id_carrera
```

```
        inner join tbl_usuario u         on u.id_usuario = d.id_docente
        inner join tbl_grupo g           on  d.id_grupo = g.id_grupo
        and g.id_carrera = d.id_carrera
        inner join tbl_horario h         on h.id_distributivo = d.id_distributivo
                inner join tbl_carrera c         on c.id_carrera = d.id_carrera
        and c.id_carrera = m.id_carrera and g.id_carrera = c.id_carrera
        and h.id_carrera = c.id_carrera
where (h.id_hora <> 0         or h.id_dia <> 0)
        and u.id_usuario = i_id_profesor
order by h.id_dia, h.id_hora;
```

Luego se almacena en otra tabla temporal el horario del profesor

```
RETURN QUERY  select * FROM crosstab('select  t1.id_hora, t1.id_dia, t2.nombre
                                    from temp_horario t1
                                    left join temp_horario_prof t2
                                    on t1.id_dia = t2.id_dia
                                    and t1.id_hora = t2.id_hora
                                    order by 1,2', 2)
                AS ct (HORA int, LUNES character varying, MARTES character varying,
                       MIERCOLES character varying, JUEVES character varying,
                       VIERNES character varying, SABADO character varying,
                       DOMINGO character varying);
END; $BODY$
LANGUAGE plpgsql VOLATILE
COST 100
ROWS 1000;
ALTER FUNCTION obtener_horario_profesor(integer)
OWNER TO postgres;
```

Acá se hace un cruce de tablas incluyendo las 2 temporales creadas anteriormente; para así tener el horario final del profesor.

Esta función contiene la lógica más compleja internamente, en el "select" inicial se encuentra el horario de un profesor, por lo que requiere la función solo de un dato de entrada. Pero esta consulta devuelve la información de una forma nada amigable con el usuario. Siendo así:

Tabla 6: Tabla de consulta a horario

Lunes	7:00 – 8:00	matemáticas I
Lunes	8:00 – 9:00	matemáticas I
Martes	8:00 – 9:00	física
Martes	9:00 – 10:00	cálculo
Miércoles	7:00 – 8:00	matemáticas I

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

En la última parte, para que el profesor pueda entender mejor su horario, se usa una función llamada "crosstab" (función perteneciente a la herramienta) que ayuda a hacer un traspaso de filas por columnas, quedando de esta forma a esta:

Tabla 7: Tabla de consulta a horario con el uso de crosstab

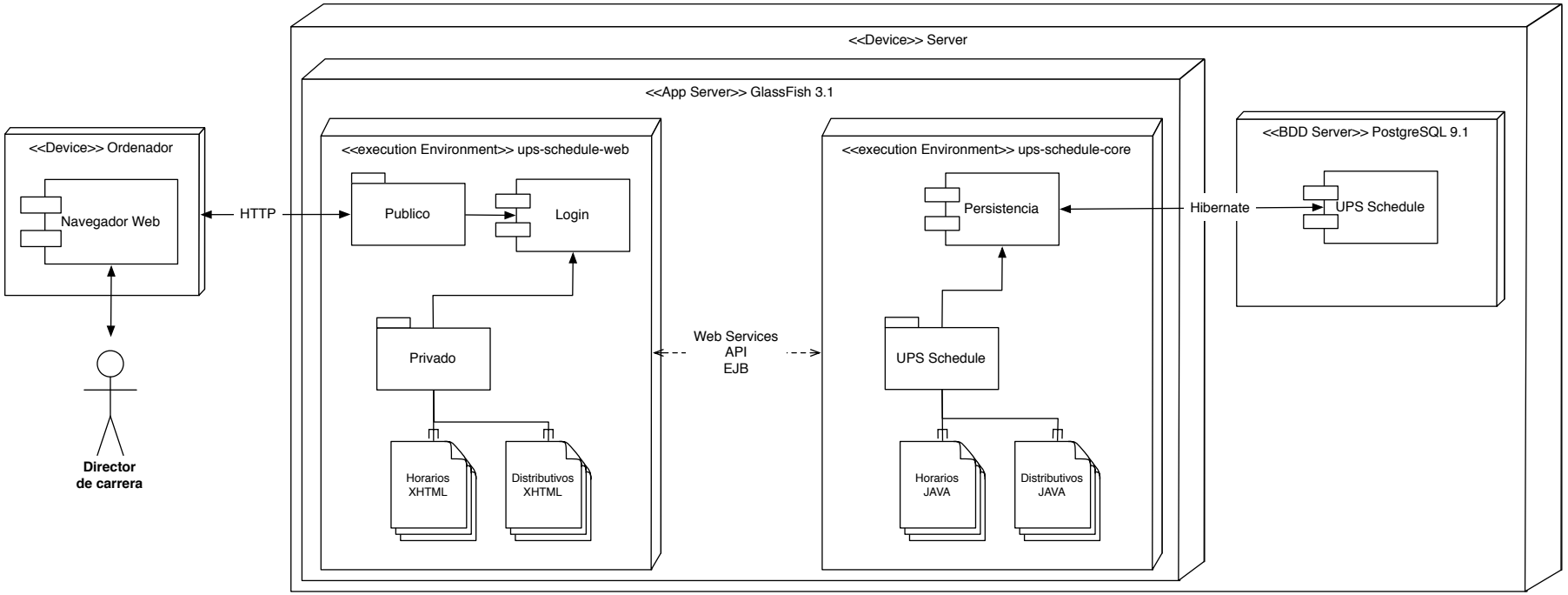
Hora	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
7:00-8:00	Matemáticas I		Matemáticas I			
7:00-8:00	Matemáticas I	Física				
7:00-8:00		Cálculo				

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

3.6. Integración

Como se menciona en capítulos anteriores el módulo de "Gestión de horarios" debe ser integrado con el módulo de "Gestión de distributivos", en la figura 39 se muestra el diagrama de integración, los procesos y actividades más importantes para realizar la tarea en mención:

Figura 39: Diagrama de despliegue, integración módulos de gestión de horarios y distributivos



Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

3.6.1. Definición de IDE

Al brindar la libertad a los grupos de seleccionar las herramientas de desarrollo para el proyecto "UPS Schedule" se generó la primera complicación, que es definir el IDE de desarrollo.

Las herramientas para el desarrollo por parte de los grupos son NetBeans y Eclipse. Tras validar las necesidades para la integración se decide migrar los archivos del desarrollo de NetBeans a Eclipse basados en:

- Eclipse brinda más libertades a los desarrolladores para analizar, ejecutar y revisar código.
- Las configuraciones en Eclipse permiten llevar un mejor control del flujo del sistema, de tal forma que facilita la detección de errores antes, durante y al final de la integración.

3.6.2. Levantamiento de servidores para la integración

Una vez definido el IDE unificado de desarrollo, se debe configurar el servidor de aplicaciones y el servidor Web con los recursos necesarios para la ejecución de los módulos.

Los principales parámetros de configuración realizados son:

- Modificación de la conexión a la base de datos y recursos del servidor, apuntando a una base de datos unificada.

```
</jdbc-connection-pool>
    <jdbc-connection-pool
        connection-validation-
method="auto-commit"
        datasource-
classname="org.postgresql.ds.PGSimpleData
Source"
        res-type="javax.sql.DataSource"
        wrap-jdbc-objects="false"
        name="postgres_bd_distribucion_pos
tgresPool">
<property
        name="URL"
        value="jdbc:postgresql://localhost
:5432/ups_schedule">
</property>
<property
        name="driverClass"
        value="org.postgresql.Driver">
</property>
<property
        name="PortNumber"
        value="5432">
</property>
<property
        name="Password"
        value="779726">
</property>
<property
        name="DatabaseName"
        value="ups_schedule">
</property>
<property
        name="User"
        value="postgres">
</property>
<property
        name="serverName"
        value="localhost">
</property>

</jdbc-connection-pool>
```

Configuración de conexión con base de datos.

3.6.3. Configuración de desarrollo

Una vez habilitada la configuración del servidor de aplicaciones, es necesario realizar la configuración del aplicativo para que permita la integración mediante:

- Instalación de librerías necesarias para la ejecución del desarrollo.
- Resolución de dependencias.

En definitiva es cargar las librerías de todos los componentes que sean necesarios, de todos sus módulos, para el correcto funcionamiento del desarrollo.

3.6.4. Cambios principales en código

Durante el proceso de integración se generaron inconsistencias que impidieron la ejecución directa de los módulos, por lo que fue necesario realizar cambios, principalmente en la forma de la conexión a la base de datos.

Se definió el uso de EJB para el manejo de la información, ya que el consumo de estos servicios facilita la transmisión de datos de forma independiente de la lógica usada dentro del módulo de gestión de distributivos y el módulo de gestión de horarios.

Adicional se puede hacer uso de la API generada en cualquier momento durante la ejecución, sin necesidad de instanciar nuevamente, lo que libera recursos en el servidor.

```
@EJB
private consulta_cruce consulta_cruce;
```

Instancia al servicio para ser usado

```
LisraDia_Hora_Cruce =
consulta_cruce.consulta_cruce_horario(o
bteneID_profesor(value),
Integer.parseInt(carrera));
```

Uso de servicio EJB

3.6.5. Conclusión

La experiencia de integrar los módulos “gestión distributivos” y “gestión horarios” ha comprobado que la combinación de desarrollo de aplicaciones en java e implementada en Glassfish fue una decisión muy viable para ejecutar este proyecto.

Al momento de integrar los módulos ya mencionados, los cambios de codificación fueron mínimos, pero se pudo comprobar la potencialidad de esta tecnología, ya que ha permitido:

- Ejecución de las herramientas multiplataforma.
- Portabilidad de las aplicaciones desarrolladas.
- Código fuente independiente de IDE de desarrollo.

CAPÍTULO 4
PRUEBAS

En este capítulo se presenta los resultados de la evaluación de las principales funcionalidades del módulo de gestión de horarios, tomando en cuenta la funcionalidad y la optimización de las respuestas brindadas (resultado y tiempo).

A continuación se muestra a detalle las principales pruebas manejadas:

4.1. Caja Blanca

Aquí se evalúa las funciones principales del módulo de gestión de horarios, ingresando los parámetros de entrada y evaluando la respuesta, tiempo de procesamiento y resultado obtenido.

Tabla 8: Tabla pruebas de caja blanca

No.	Función	Entrada	Tiem	Resp	Observaciones
1	obtener_horario_profesor	Integer, Id_profesor	790 ms	OK	Resultado esperado
2	actualiza_horario	Integer, id_horario Integer, id_dia Integer; id_hora	10 ms	OK	Resultado esperado
3	obtener_materias	Integer, id_carrera Integer, id_nivel Integer; id_grupo	18 ms	OK	Resultado esperado
4	obtener_materias_sin_horario	Integer, id_carrera Integer, id_nivel Integer; id_grupo	18 ms	OK	Resultado esperado
5	obtener_niveles	Integer; id_carrera	21 ms	OK	Resultado esperado
6	obtener_grupo	Integer, id_carrera Integer, id_nivel	20 ms	OK	Resultado esperado
7	obtener_profesores	Integer, id_carrera Integer, id_nivel Integer; id_grupo	18ms	OK	Resultado esperado

No.	Función	Entrada	Tiem	Resp	Observaciones
8	obtener_cruce_profesor	Integer, id_profesor Integer, id_carrera	134 ms	OK	Resultado esperado
9	cambiarcolorfondo	Integer, id_objeto String, color	2 ms	OK	Resultado esperado

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

4.2. Caja Negra

En esta prueba se evalúa los procedimientos que el usuario realiza, que los pasos necesarios se encuentren funcionando correctamente y que el resultado de dichos procedimientos sea satisfactorios.

Tabla 9: Tabla pruebas de caja negra

No.	Nombre	Req	Pasos	Resp	Observaciones
1	Generación de horarios	1.1	Seleccionar Horarios Seleccionar Generar	OK	Resultado correcto, presentación de materias asignadas y no asignadas.
2	Modificación de horarios	1.2	Seleccionar materia Mover materia Asignar nuevo horario	OK	Las funciones de mouse y la actualización de cambio en la base de datos, funcionan correctamente.
3	Validación de cruce en horario	1.3	Seleccionar materia Mover materia	OK	La validación de cruces y cambio de colores funcionan correctamente.
4	Reportes de horario de profesores	1.4	Seleccionar reportes Seleccionar horario Seleccionar profesor Ingresar profesor	OK	
5	Reporte de horario por grupo	1.5	Seleccionar reportes Seleccionar horario Seleccionar grupo Ingresar grupo	OK	Presenta el horario de un solo profesor a la vez.

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

4.3. Estrés

Las pruebas de estrés permiten evaluar el desempeño del módulo de gestión de horarios y sus funciones, con respecto al rendimiento, a través de JUnit para las pruebas de la lógica de la aplicación y pgAdmin para la lógica de negocio.

Tabla 10: Tabla pruebas de estrés

No.	Detalle	Carga	Respuesta	Observaciones
1	Persistencia de aplicación	150 usuarios	El rendimiento del servidor de aplicaciones, con una carga proyectada a 4 veces la carga real es satisfactoria.	Respuesta del servidor, satisfactoria.
2	Persistencia de base de datos	80 funciones	El tiempo de respuesta de la base de datos no supera los 300 ms en las funciones más complejas. En cuanto al rendimiento del motor de base de datos no supera del 20%.	El manejo de funciones optimiza el rendimiento las consultas.
3	Funcionalidad del sistema en diferentes navegadores	4 navegadores	El rendimiento es el mismo respecto al funcionamiento del sistema.	El tiempo de respuesta en cada navegador varia; pero no es notorio al usuario.
4	Consulta cruces	50 usuarios	Respuesta correcta en consulta para cada usuario.	Tomar en cuenta el cruce se efectúa en cada clic, si es movido el horario durante el cruce no se refleja hasta dar otro clic.
5	Consultar horario de grupo	50 usuarios	Carga horario correctamente.	El tiempo de respuesta varía dependiendo del computador del usuario.
6	Actualizar horario	100 usuarios	Actualización correcta.	Las actualizaciones son inmediatas ya que se la hace en base a su id.
7	Servidor de base de datos fuera de línea	1	La aplicación no funciona, envía a página de error de sistema	En cuanto el servidor de base de datos está en línea nuevamente el sistema sigue su funcionalidad normalmente, sin necesidad de reiniciar al servidor web.

Elaborado por: Oviedo Guallichico Danilo Román & José Antonio Granda León

CONCLUSIONES

- Se ha obtenido un sistema para directores de carrera, integrando los módulos de gestión de distributivos y gestión de horarios, totalmente funcional con tecnologías nuevas, que facilitan notablemente la generación de horarios.

- Queda de mostrado que el uso de JAVA es una excelente opción para desarrollo de sistema, ya que durante el desarrollo (IDE NetBeans) e integración (IDE Eclipse) se demostró la compatibilidad entre los diferentes IDE y portabilidad, no fue un impedimento para desarrollar en una plataforma WIN 32, WIN 64, MAC OS 32, MAC OS 64.

- Durante las pruebas se evidenció que el rendimiento del servidor de aplicaciones Glassfish 3.1 y el servidor de base de datos PostgreSQL 9.1 supera las expectativas de los requerimientos de rendimiento solicitadas.

- Aplicar las reglas de negocio a nivel de base de datos en PostgreSQL 9.1 permitió mejorar el rendimiento de la aplicación de forma notable, adicional a la facilidad que brinda en el mantenimiento cuando se aísla de la lógica de la aplicación.

- La metodología OMT es una excelente opción para el desarrollo grupal de aplicaciones ya que busca modelar claramente resolver el problema y las funciones necesarias sin enfocarse en el desarrollo en sí, pero es necesario tomar en cuenta que existen muchas variaciones nuevas de OMT que agilitan y facilitan el modelar la nueva solución.

RECOMENDACIONES

- Se recomienda el uso de EJB para evitar la instanciación continua de los objetos de un sistema.

- Se recomienda el uso de IDE NetBeans si el desarrollador es novato, por que permite configuraciones sencillas y una vez la experiencia aumente usar el IDE Eclipse que permite configuraciones mucho más personalizables.

- Se recomienda explotar al máximo el motor de la base de datos para las funciones de la lógica de negocio, ya que los motores de base de datos poseen un gran trabajo de ingeniería en la búsqueda de la optimización de tiempos de respuesta.

- Se recomienda revisar las metodologías que han evolucionado de OMT para el desarrollo de software, en búsqueda de la optimización de tiempos de desarrollo y resultados.

- Es recomendable el uso de Jasper Report en lugar de iReport debido a que las funcionalidades de iReport no son totalmente multiplataforma.

- Se recomienda el uso de PrimeFaces que permite un desarrollo muy sencillo de interfaces muy elaboradas.

LISTA DE REFERENCIA

Universidad Politécnica Salesiana. (2009). *Sistema Nacional Académico.* Quito, Pichincha, Ecuador.

Yakov, F. (2010). *Programación Java 1.* España: ANAYA MULTIMEDIA/WROX.

Aguilar, L., & Martinez, I. (2011). *Programación en JAVA 6 algoritmos, programación orientada a objetos e interfaz gráfica de usuario.* México: Mac Graw Hill.

Bonilla, G. (2007). *Como hacer una tesis de graduación con técnicas estadísticas* (3ra Edición ed.).

Deitel, P., & Deitel, H. (2012). *JAVA Como Programar* (9na Edición ed.). México.

Fuentes, J. M. (2009). *Manual de Ajax, Las entrañas de Ajax.* Retrieved 4 de 2012 from El rincon de ajax: http://www.elrincondeajax.com/wp-content/uploads/Manual.pdf

Libros Web. (n.d.). *Libros Web.* Retrieved 7 de 2012 from Libros Web: http://www.librosweb.es/ajax/

Linkaterra. (n.d.). *video tutoriales principalmente alojados en YouTube*. Retrieved 3 de 2012 from Linkaterra: http://www.linkaterra.net/

Óros, J. C. (2012). *Diseño de paginas Web con XHTML, JavaScript y CSS* (3ra Edición, ed.). México: Alfaomega.

Ortiz, M., & Plaza, A. (2013). *Programación orientada a objetos con JAVA y UML* (1ra Edición ed.). Ecuador: Editorial Universitaria Abaya-Yala.

PrimeFaces. (n.d.). *Demos y ejemplos, Código abierto*. Retrieved 2 de 2012 from Página Oficial PrimeFaces: http://www.primefaces.org/showcase/ui/home.jsf

PrimeFaces. (n.d.). *Manual de Usuario Oficial de librería Prime Faces*. Retrieved 2 de 2012 from PrimeFaces USER'S GUIDE: http://www.primefaces.org/documentation.html

Stinson, B. *PostgreSQL essential reference* (1ra Edición ed.). 2009: New Riders.

Rumbaugh, J. (1996). *Modelado y diseño orientado a objetos.* et al.

Tufiño, R. E. (2012). *Sistema UPS Schedule para la Universidad Politécnica Salesiana Sede Quito.* Quito, Pichincha, Ecuador.

ANEXOS